VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NEEUKLIDOVSKÉ VYKRESLOVÁNÍ VE VR

NON-EUCLIDEAN RENDERING IN VR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MATEJ BOBUĽA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Bobuľa Matej, Bc.**
Program: Informační technologie
Obor: Počítačová grafika a multimédia
Název: **Neeuklidovské vykreslování ve VR**
Non-Euclidean Rendering in VR
Kategorie: Počítačová grafika
Zadání:

1. Nastudujte techniky VR a možnosti vykreslování neeuklidovských prostorů.
2. Navrhněte VR aplikaci využívající vybrané techniky neeuklidovského zobrazování.
3. Implementujte navrženou aplikaci.
4. Zhodnoťte a proměřte implementovanou aplikaci.
5. Vytvořte demonstrační video.

Literatura:

- Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Milet Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cieľom tejto diplomovej práce je naštudovať si možnosti vykresľovania geometrie a priestorov vo virtuálnej realite. Zoznámiť sa s pojmom neeuklidovskej geometrie a neeuklidovského priestoru, jej vznikom, ako aj s princípmi, ktoré sa využívajú v hernom priemysle na ich simuláciu. Na základe naštudovaných materiálov potom vhodne zvoliť vývojové prostredie v ktorom sa samotná aplikácia, demonštrujúca takýto priestor, bude realizovať a implementovať zvolené techniky za pomoci ktorých sa dosiahne neeuklidovský priestor, efekt. Výsledkom práce je spustiteľná aplikácia navrhnutá na Windows platformách. Principiálne sa jedná o počítačovú hru, kde cieľom hráča je prechádzať jednotlivé úrovne, kde individuálne úrovne boli vytvorené tak, aby demonštrovali vždy mierne odlišný neeuklidovský priestor, efekt.

Abstract

The main goal of this master's thesis is to research different approaches of rendering geometries and spaces in virtual reality. Learn more about the terms, non-Euclidean geometry and non-Euclidean spaces, their origin and different principles used in video game industry to simulate such geometries or spaces. Based on the research, a selection of an optimal API is needed for the implementation of such application. Application is designed to run on desktop computers with Microsoft Windows operating system. Application, in it's core, is a video game and the main goal of the player is to successfully complete each and every level of the game. These levels are designed in a specific way so that they each individually represent some form of non-Euclidean geometry or space.

Kľúčové slová

virtuálna realita, VR, Euklidovské priestory, neeuklidovské priestory, herný engine Unity, Unity 3D, viacpriechodové vykresľovanie, vykresľovanie do textúr, portály, stencilový buffer, stencilová maska, stereoskopické vykresľovanie

Keywords

virtual reality, VR, Euclidean spaces, non-Euclidean spaces, game engine Unity, Unity 3D, multi-pass rendering, render to texture, portals, stencil buffer, stencil mask, stereoscopic rendering

Citácia

BOBUĽA, Matej. *Neeuklidovské vykreslování ve VR*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet

Neeuklidovské vykreslování ve VR

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením inžiniera Tomáša Mileta. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matej Bobuľa
18. mája 2021

Poďakovanie

Srdečne by som sa v prvom rade chcel poďakovať Ing. Tomášovi Miletovi, ktorý mi v priebehu celého riešenia práce poskytoval odborné vedenie, užitočné rady, nové nápady a dojmy. Ďalej by som rád vyslovil vďaku aj svojej rodine a blízkym za psychickú podporu.

Obsah

Kapitola 1

Úvod

Vývoj počítačových hier ako aj nových zariadení a spôsobov ako dané hry zažiť je v dnešnej dobe čoraz viac a viac. Na dnešnom trhu sa môžeme stretnúť s rôznymi typmi hier ako sú akčné strielačky, adventúry, strategické hry, športové hry, hry na hrdinov a mnoho iných. S rôznymi žánrami hier súvisia aj periférne zariadenia, ktoré vznikali za účelom obohotenia herného zážitku z hry, ponorenia hráča do sveta hry. Od základných periférií ako klávesnica, myš a vymyslenia herných joystickov na letecké simulátory, volantu a pedálov na hranie závodných hier, gamepadov i ďalších, bolo len otázkou času kedy vznikne nový spôsob, ktorý upúta pozornosť populácie a odštartuje „novú éru hrania hier“. Stalo sa tak s príchodom virtuálnej reality.

Virtuálna reality a dopyt po nej za posledné roky náramne vzrástol. Hoci je to pomerne starý termín, technológia umožňujúca prenos človeka do čisto virtuálneho sveta je pomerne nová. V kapitole 2 bude čitateľ tejto práce oboznámený s krátkou históriou virtuálnej reality, dvomi mierne odlišnými koncepciami, ktoré položili základy chápania virtuálnej reality a jej realizácie. Ďalej budú v kapitole popísané dnes najpopulárnejšie zariadenia, ktoré formovali a ďalej formujú smer vývoja virtuálnej reality. Primárnymi zdrojmi popisu obecnej histórie sa stali články [9] a [1]. Na popis jednotlivých zariadení poslúžili oficiálne stránky ich výrobcov.

Podobne ako virtuálna realita predstavuje novú koncepciu hrania hier tak aj neeuklidovský priestor predstavuje nový spôsob chápania svojho okolia. V úvodných častiach kapitoly 3 si zaspomíname na školské ľavice, v ktorých sme sa po prvý krát stretli s Euklidovským priestorom, objasníme si čo pod daným termínom môžeme rozumieť a ako z daného priestore zostrojiť neeuklidovský. Ako príklad si ďalej uvedieme zopár aplikácií vykresľujúcich neeuklidovský priestor po matematickej stránke a taktiež si popíšeme spôsoby a aplikácie, ktoré sa snažia prepojiť práve takýto priestor s herným prostredím.

Na samotný návrh a tvorbu aplikácie bol použitý herný engine Unity. S popisom čo daný engine umožňuje, aké platformy podporuje a aké sú základné koncepty práce v ňom, sú popísané v kapitole 4. Čitateľ bude ďalej oboznámanený zo samotným editorom enginu, v ktorom sa odohráva väčšina práce s ním a princípy akými je možné v tomto engine vykresľovať prostredia virtuálnej reality. V druhej polovici kapitoly bude čitateľ oboznámený s procesom tvorby samotnej aplikácie. Popísané budú zvolené metódy, ktoré boli implementované na dosiahnutie neeuklidovského priestoru, pohyb hráča takýmto priestorom a návrh aplikácie, či už užívateľského rozhrania alebo grafického prevedenia.

Práca je zakončená jednoduchým vyhodnotením aplikácie v kapitole 5 a v kapitole 6 sú popísané rôzne návrhy na rozšírenie aplikácie, či už v oblasti kompatibility s iným hardvérom alebo samotnej funkcionality, dizajnu.

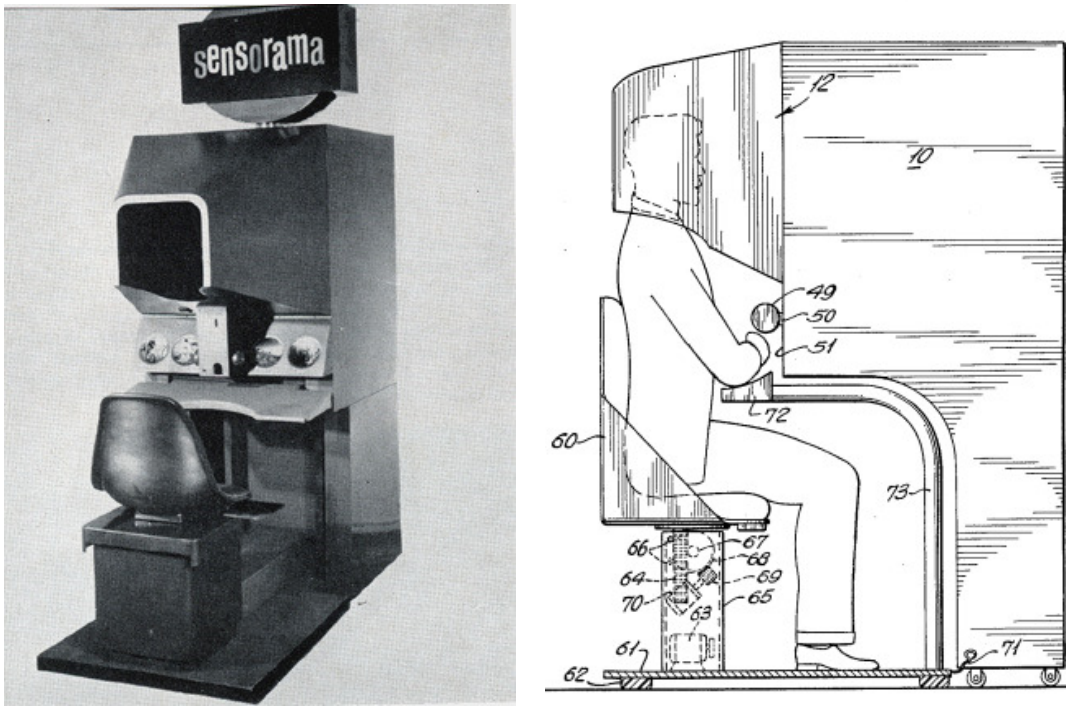
Kapitola 2

Virtuálna realita

Samotný pojem „virtuálna realita“ existuje už pomerne dlhý čas. Nemá presne stanovený dátum vzniku z dôvodu problému chápania rôznych koncepcií a interpertácií čo všetko má zahŕňať, čo to vlastne je. Zmienky o virtuálnej realite siahajú až do 18. storočia, avšak jednou z najznámejších sa snáď stala zmienka zo science fiction knihy z roku 1935 od spisovateľa menom Stanley G. Weinbaim. Tento americký spisovateľ vo svojej knihe, resp. krátkom príbehu „*Pygmalion's Spectacles*“, vytvoril postavu Alberta Ludwiga, elfského profesora, ktorý zostrojil sadu okuliarov umožňujúce nositeľovi zažiť „film dávajúci danému zrak a zvuk, ... chuť, čuch a hmat“. Práve tento koncept „nasaď si okuliare a nechaj sa preniesť kam sa ti myseľ zatúla“ sa stal veľmi populárnym a dalo by sa povedať, že položil stavebné bloky pre virtuálnu realitu ako ju poznáme dnes. V dnešnej dobe sa virtuálna realita stáva čoraz viac a viac prístupnejším zariadením pre šíre publikum ako po cenovej stránke tak aj po stránke hardvérovej. V nasledujúcich podkapitolách je popísaný vývoj týchto zariadení a následne aj najnovšie a najzaujímavejšie zariadenia umožňujúce bežnému užívateľovi poskytnúť práve zapojenie sa do virtuálnej reality.

2.1 História

Tak ako už bolo spomínané v úvode kapitoly, nie je možné úplne presne stanoviť počiatky virtuálnej reality. Jednalo sa o veľmi pokročilý koncept, ktorý si v minulom storočí vedelo predstaviť nie veľa ľudí. Rôzne výklady, čo má daný pojem obnášať viedlo k rôznym ľudským výtvorom. Jednotlivé pokusy o zostrojenie takéhoto prístroja sa od seba spočiatku teda veľmi odlišovali. Úplne prvé hmotné implementácie virtuálnej reality sa začali objavovať v päťdesiatych a šesťdesiatych rokoch minulého storočia. Za týmito pokusmi stála skupinka dnes neznámych ľudí až dokým si v roku 1962 filmový producent, režisér a spisovateľ Morton Leonard Heilig [19] nedal patentovať zariadenia pod názvom „*Sensorama*“. Jednalo sa o kabinet v štýle klasických arkádových hier. Na rozdiel od nich však *Sensorama* obsahoval stereoskopický obraz, čo dávalo dojem 3D obrazu, vibrujúce kreslo, kabinet bol schopný fúkať vzduch do tváre ako aj produkovať rôzne pachy mesta. Zariadenie je možné vidieť na obrázku 2.1.



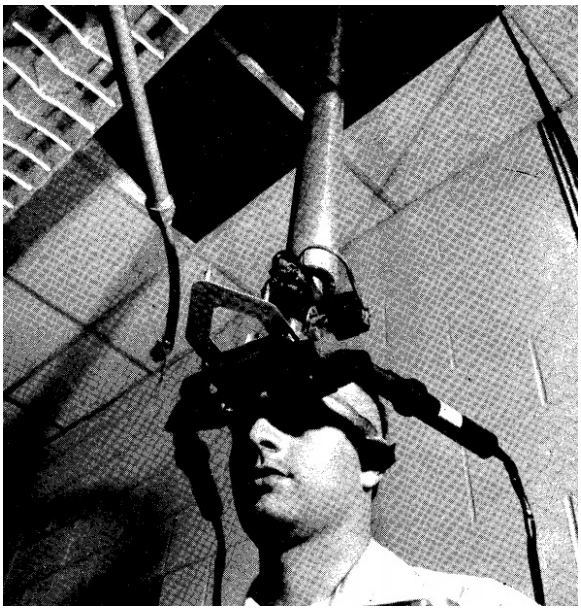
Obr. 2.1: V ľavo kabinet „Sensorama“, v pravo príklad použitia priamo z patentu [4].

Neskôr v roku 1965 Ivan Edward Sutherland, americký profesor ocenený Turingovou cenou [1], vo svojom krátkom článku „The Ultimate Display“[12] popísal ako si on predstavuje dokonalé počítačové zobrazovanie. Popisuje, že podľa jeho predstáv by sa jednalo o miestnosť v ktorom by samotný počítač bol schopný ovládať a manipulovať s hmotou. Opúšťa sa predstavy o tom, že zariadenie by bolo alebo malo byť schopné produkovať chute a pachy. V článku sa zároveň zmieňuje aj myšlienka, že samotné objekty v takto „dokonalom“ zobrazení by nemuseli nasledovať jednoduché pravidlá fyziky a matematiky. Toto zariadenie by teda slúžilo len ako zrkadlo do sveta skonštruovaného v pamäti počítača. S prvou hmotnou iteráciou tejto svojej vízie o „dokonalom zobrazovaní“ prišiel v roku 1968 [13]. Jednalo sa o náhlavný displej (anglicky „Head-Mounted Display“, skratka HMD) pripomínajúci periskop, viď obrázok 2.2. Zariadenie pozostávalo s CRT displejov, čo ho nerobilo akurát najľahším a tak musel byť vešaný zhora (obrázok 2.3) aby ho bolo možné používať. Taktiež si vďaka svojej váhe a spôsobu zapojenia vyslúžil prezývku „Damoklov meč“. Tento headset do istej miery spĺňoval autorovu víziu, keďže bol schopný kresliť primitívne telesá pomocou čiar, bol však značne limitovaný technológiou svojej doby.

[1] cena Alana M. Turinga, povazovaná za najvyššiu možnú cenu v oblasti počítačovej vedy

Obr. 2.2: Damoklov meč, z boku a z hora, prebraté z [8].

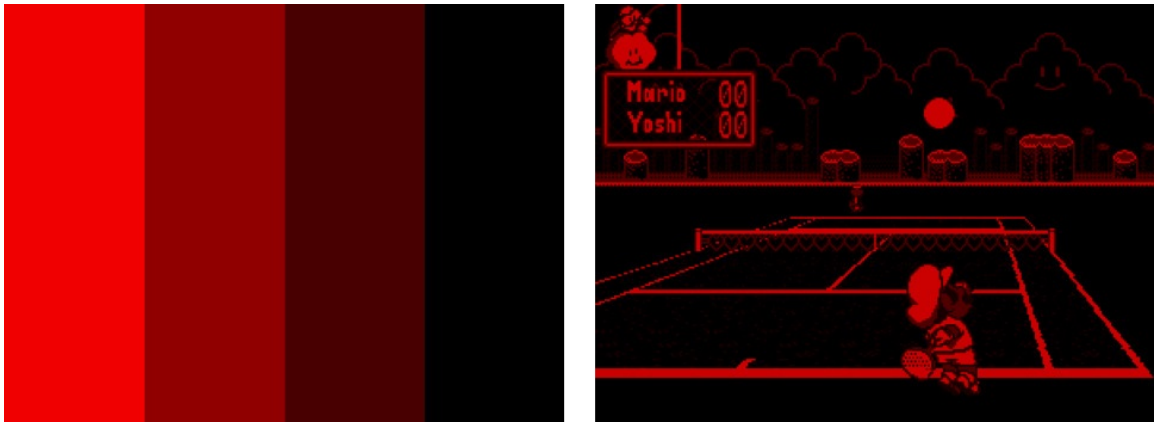
V sedemdesiatych a osemdesiatych rokoch vývoj zariadení virtuálnej reality pokračoval ďalej. Primárne sa však vyvíjali drahé zariadenia, ktorých využitie bolo v armáde a medicíne. Za zmienku stoja zariadenia ako systém LEEP (Large Expense, Extra Perspectie), ktorý neskôr Nasa transfomovala na headset s názvom VIEW.



Obr. 2.3: Damoklov meč v akcií.

Prvé verejne použiteľné headsety sa objavili až keď sa k idei virtuálnej reality dostali herné štúdiá v deväťdesiatych rokov. Jedným z popredných herných štúdií boli japonské SEGA a Nintendo. Sľubovali revolučný spôsob hrania hier aký doteraz nikto nevidel. V prípade headsetu od štúdia Sega, Sega VR, boli oznámené tri verzie. Jeden headset pre arkádové hry a dva headsety pre domáce konzoly Sega Genesis a Sega Saturn. Svetlo sveta uzrela nakoniec však len arkádová verzia a zvyšné dva boli úplne zrušené. V prípade headsetu „Virtual Boy“ od Nintenda, ktorý vyšiel o dva roky neskôr v 1993 to dopadlo rovnako biedne. Hneď prvým a najzávažnejsím problémom bolo, že sa prakticky vôbec nejednalo o headset ale o okuliare na stojane, viď obrázok 2.4. To spôsobovalo veľmi nepraktický spôsob hrania hier. Ďalším závažným problémom bol nedostatok hier. Dokopy bolo vydaných 22 hier, z toho veľa bolo dostupných len v Japonsku a ako keby to nestačilo, samotné hry boli síce 32-bitové, no Virtual Boy podporoval len tri odtiene červenej farby plus čiernu, obr. 2.5. Po tomto neúspechu sa herné spoločnosti dlhšiu dobu radšej virtuálnej realite vyhýbali.

Obr. 2.4: Virtual Boy „headset“. Jeho dizajn umožňoval len veľmi náročné sa vnorenie (anglicky „immersion“) do hry. Pri každej zmene polohy hráča alebo len náhodného postrčenia headsetu došlo okamžite k narušeniu spomenutého vnorenia.



Obr. 2.5: Naľavo podporovaná farebná paleta, napravo jej využitie v hre Mario Tennis.

Po takmer dvoj-dekádovej odstávke nastal zhruba v roku 2012 obrat vo vnímaní virtuálnej reality. Jednalo sa o rok kedy bol na webovej stránke spoločnosti Kickstarter[2] po prvý krát zverejnený headset Oculus Rift 2.2. Jeho autorom bol vtedy iba 22-ročný Palmer Freeman Luckey. Uvedenie tohoto headsetu na trh opäť naštartoval záujem používať virtuálnu realitu. Všetci popredný výrobcovia technológií a hier si začali vyvíjať vlastné headsety a aplikácie na rôzne platformy, kompatibilné s rôznymi zariadeniami, či už sa jednalo o stolové počítače, notebooky, tablety alebo smartfóny. V dnešnej dobe si môže už takmer každý stiahnuť na svoj smartfón VR aplikáciu a na jej sledovanie, hranie, postačí vlastnoručne vyrobený headset napríklad aj z kartónového papiera ako je Google Cardboard, obrázok 2.6.

[2]ide o spoločnosť podporujúcu nove nápady a startup projekty, dostupné na https://www.kickstarter.com/

Obr. 2.6: Google Cardboard.

2.2 Oculus

Ako už bolo spomínané Oculus bol jedným z prvých moderných dostupých VR headsetov. Počas svojho vývoja mal niekoľko verzií, prototyp z roku 2010 a následne dve ďalšie verzie pomenované DK1 (2012) a DK2 (2014) (z anglického „Development Kit“ 1 a 2). Novšie iterácie headsetu nezlepšovali len jeho kvalitu ale zlepšovali aj uživateľský komfort. Oficiálne prvá verejná verzia, Oculus Rift z roku 2016 [14], mala nasledovné špecifikácie:

- dva OLED displeje v rozlíšení 1080x1200 (celkove 2160x1200)
- obnovovaciu frekvenciu 90 Hz
- šírka zorného poľa 110 stupňov

Neoddeliteľnou súčasťou balenia bol aj senzor 2.7 na sledovanie polohy headsetu a neskôr aj na sledovanie iných VR zariadení, ovládačov. Senzor zachytáva infračervený lúč led diód umiestnených v headsete a na základe nich potom určuje pozíciu vo virtuálnom svete. S počiatku samotné balenie obsahovalo iba jeden ovládač Xbox One od spoločnosti Microsoft. Oculus Rift bol totiž navrhnutý tak, že pôvodne senzor podporoval iba sledovanie headsetu v uhle 180 stupňov. Predpokladalo sa totiž, že užívateľ si spustí VR aplikáciu podobne ako hocijakú inú počítačovú hru, v sede za počítačom. Toto však nebolo práve veľmi dobré riešenie pretože pohyb vo virtuálnej realite pomocou takéhoto ovládača spôsobuje kinetózu, viď kapitola 2.6. Nástupcom sa stali ovládače Oculus Touch, obrázok 2.8, ktoré po svojom obvode majú nainštalované rovnaké infračervené led diódy ako headset, takže ich bolo možné rovnako sledovať vo VR. Senzor sa zapája do PC pomocou USB 2.0/3.0 a headset pomocou USB 2.0/3.0 a HDMI vstupu.

Obr. 2.7: Zľava doprava: senzor, headset Oculus Rift, Xbox One ovládač.

Obr. 2.8: Ovládače Oculus Touch.

Z postupom času sa i táto verzia Oculusu postupne zdokonaľovala. Dnes máme Oculus dostupný v dvoch verziách Oculus Quest 2 a Oculus Rift S.

2.2.1 Oculus Quest 2

Predstavuje variantu pre menej náročných užívateľov. Je určená nielen pre stolové počítače ale aj pre smartfóny. Samotný headset obsahuje procesor, vlastnú pamäť a úložný priestor. Párovanie medzi smartfónom a headsetom prebieha pomocou bluetooth. Najdôležitejšie parametre headsetu sú nasledovné:

- procesor Qualcomm Snapdragon XR2
- 6 Gb RAM
- 6 až 256 Gb úložného priestoru na hry
- jeden LCD displej v rozlíšení 3664x1920 (1832x1920 na jedno oko)
- obnovovaciu frekvenciu 72 Hz - 90Hz
- šírka zorného poľa 90 stupňov

2.2.2 Oculus Rift S

Oculus Rift S predstavuje vylepšenú verziu pôvodného Oculus Rift. Podobne ako jeho predcháczajúca verzia je primárne určená pre náročnejších užívateľov. K tomuto headsetu je nutný počítať minimálne s procesorom **Intel i3-6100** alebo **AMD Ryzen 3 1200**, grafickou kartou **NVIDIA GTX 1050 Ti** alebo **AMD Radeon RX 470**, **8Gb RAM**[15]. Zariadenie je možné pripojiť cez mini DisplayPort (skratka mDP), odporúčaný je však klasický DisplayPort. Špecifikácie headsetu boli po každej stránke vylepšené a sú nasledovné:

- jeden LCD displej v rozlíšení 2560x1440 (1280x1440 na jedno oko)
- obnovovaciu frekvenciu 80 Hz
- šírka zorného poľa 90 stupňov

Keďže je headset určený pre stolové pc je jasné, že po zobrazovacej stránke dosahuje značne vyššej kvality ako Quest 2. Ovládače sú v oboch headsetoch odvodené od pôvodných Oculus Touch ovládačov s novým spôsobom ich sledovania, zlepšenou životnosťou batérie a drobnými estetickými zmenami.

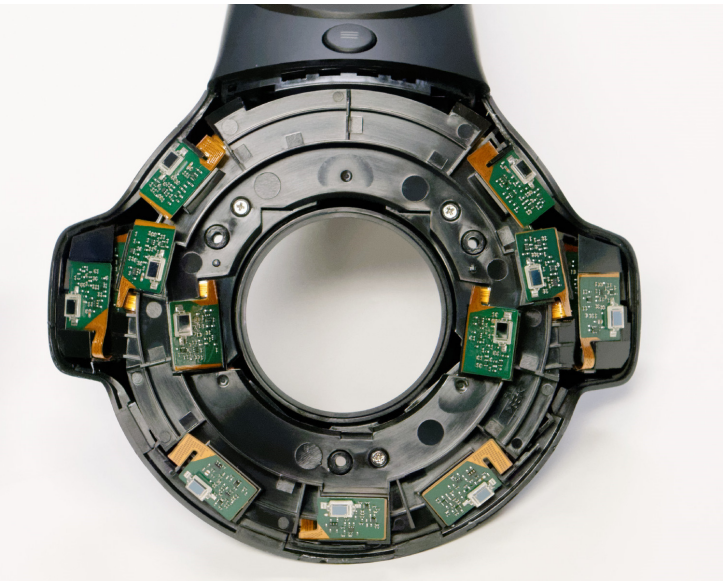
2.3 HTC Vive

Snáď druhým najrozšírenejším headsetom po Oculus-e a jeho hlavným konkurentom sa stal headset od spoločnosti HTC a Valve. Na trh prišiel v rovnakom roku ako prvý Oculus Rift, v roku 2016. Zo svojím príchodom však samotný headset priniesol zopár nových vlastností oproti konkurencií. Tento headset ako prvý podporoval **room-scale** VR. Znamenalo to že oproti pôvodnému Oculusu Rift, bolo možné si v rámci virtuálnej reality definovať hrací priestor $5m^2$, a pohyb v reálnom svete bol odzrkadlený do virtuálneho sveta.

Zo snahou dosiahnuť stanovený návrh room-scale VR bolo nutné vymyslieť ovládače, ktoré by bolo možné sledovať v rámci tohoto návrhu. Inšpiráciou pre ne sa stali už existujúce ovládače, PlayStation Move spomenutý v kapitole 2.4 a Steam Controller, od firmy Valve.

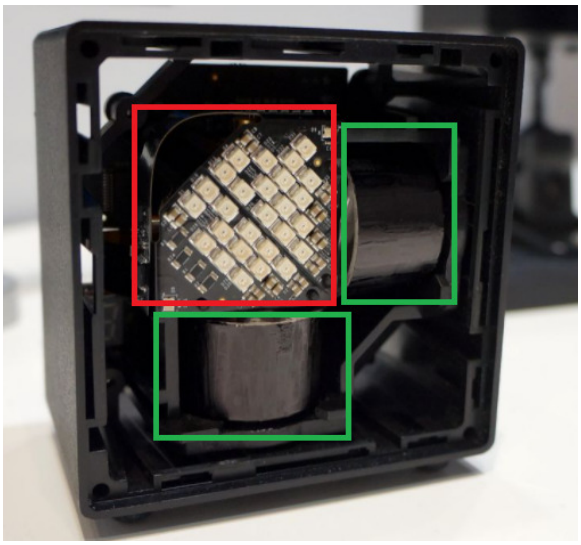
Ovládače po obvode obsahujú 24 senzorov pomocou ktorých sme shcopný určiť ich polohu a rotáciu vo VR na desatiny milimetra [2]. Na obrázku 2.9 je možné vidieť ako vyzerá

rozobratý ovládač z odkrytými senzormi a na obrázku 2.11 kompletnú zostavu headsetu z ovládačmi a majákom.



Obr. 2.9: Odkryté senzory z hora na ovládači HTC Vive, prebraté a upravené z [7].

Samotné sledovanie zariadení je zabezpečené systémom Lighthouse vo verzii 1.0. Za vznik vďačíme jeho vynálezcovi menom Alan Yates. Meno Lighthouse (slovensky „maják“) je pre tento systém, veľmi výstižné, keďže zariadenia obsahuje dva rotačné motory, ktoré strieľajú laserové lúče po miestnosti. Podobne ako ovládače tak i headset obsahuje senzory na detekciu laseru. Samotná pozícia a rotácia zariadenia je zistená na základe času medzi detekciou vyslaného lúča a detekciou synchronizačného bliknutia emitorov [3]. Na obrázku 2.10 je Lighthouse 1.0, zelenou farbou sú vyznačené rotačné motory vysieľajúce laserové lúče do okolia, červenou farbou sú vyznačené emitory synchronizačných pulzov, blikajúce vždy keď sa rotory nachádzajú v určitej polohe.



Obr. 2.10: Majáky „Lighthouse“ vo verzii 1.0.

Obr. 2.11: Kompletná zostava HTC Vive, vzadu dva majáky Lighthous vo verzii 1.0, ovládače HTC Vive a samotný headset.

Okrem samotného HTC Vive z roku 2016 je dostupná aj jeho vylepšená verzia HTC Vive Pro z roku 2018. Najvätšou zmenou oproti klasickému Vive bolo väčšie rozlíšenie očných displejov. Na jedno oko má rozlíšenie až 1440x1600, celkovo má teda headset displeje v rozlíšení 2800x1600. V kombinácií s modernými displejmi umožňujúcimi až 615 PPI (z anglického Pixels Per Inch, v preklade Pixely na palec), bolo odhadované zlepšenia zhruba 78%. Špecifikácie HTC Vive z rou 2016:

- dva OLED displeje v rozlíšení 1080x1200 (celkove 2160x1200)
- obnovovaciu frekvenciu 90 Hz
- šírka zorného poľa 100-110 stupňov

2.4 PlayStation VR

Palmer Lucky nebol jediný koho fascinovala virtuálna realita. Vývojári od Sony Interactive Studios na nej pracovali zhruba od 90. rokov avšak pomerne neúspešne. Najvätší prielom, ktorý spoločnosť zažila v oblasti VR, bolo vydanie ovládača ktorý vyvinul Richard Marks spolu zo svojím týmom, ovládač PlayStation Move [6]. Tento ovládač umožňoval užívateľovi ovládať konzolu na základe pohybu. Ovládač obsahuje na svojom vrchu svietiacu guľu. Je to z dôvodu aby kamera, ktorá ho sleduje, bola schopná daný ovládač v reálnom svete nájsť. Sledovanie ovládača za pomoci kamery nerobil žiadny problém po horizontálnej a vertikálnej osi. Problém bol ako zistiť vzdialenosť od kamery. Tu našla táto guľa ďalšie využitie, pretože vzdialenosť ovládača od kamery je vypočítaná na základe pôvodného priemeru tejto gule a priemeru, ktorý kamera zachytáva. Samotný ovládač nezožal veľký úspech, na trh prišiel pomerne neskoro, zhruba 4 roky po vydaní ovládača Wii Remote. Ovládač však sám o sebe fungoval veľmi dobre a samotný vývojáry si čoskoro uvedomili že princíp, ktorým sú schopný sledovať jednotlivé ovládače môže byť aplikovaný aj na iné zariadenia. Na obrázku 2.12 je možné vidieť samotné ovládače PlayStation Move a prototypy ich „iného“ využitia.

Obr. 2.12: Naľavo ovládače PlayStation Move, napravo snaha o ich integráciu do headsetu.

PlayStation VR vyšlo taktiež v roku 2016 a umožňoval sledovanie o uhle 360 stupňov. Na sledovanie slúži PlayStation kamera a keďže samotný headset je od firmy SONY je možné ho bohužiaľ používať iba z ich poprednými konzolami PlayStation 4 a PlayStation 5. Ovládanie a pohyb vo virtuálnej realite je dosiahnutý za pomoci ovládačov PlayStation Move. Parametre headsetu sú nasledovné:

- dva OLED displeje s rozlíšením 1920x1080 (960x1080 na jedno oko)
- obnovovaciu frekvenciu 120Hz a 90 Hz
- šírka zorného poľa 100 stupňov

2.5 Valve Index

Po úspešnom vydaní headsetu HTC Vive, ktorý vznikol kolaboráciou Valve a HTC, sa Valve rozhodol vydať vlastnou cestou. V roku 2019 nielenže oznámili ale zároveň rovno aj vydali vlastný headset s názvom Valve Index[20]. Headset po každej stránke vylepšuje vlastnosti headsetu HTC Vive. Zlepšenie samotného headsetu však nebola jediná inovácia s ktorou Valve prišiel. Snažili sa držať svojej idei **room-scale** VR. Priniesli tak vylepšenú verziu sledovania, Lighthouse 2.0, ktorá umožňuje si definovať až $10m^2$ hernej zóny namiesto pôvodných $5m^2$. Horizontálne zorné pole majákov bolo rozšírene zo 110 stupňov na 150 stupňov. Namiesto dvoch rotačných motorov na vysielanie lúčov má Lighthouse 2.0 iba jeden rotačný motor a synchronizácia je implementovaná za pomoci týchto lúčov, namiesto emitorov synchronizačných pulzov. Toto umožňuje spárovať až štyri zariadenia Lighhouse 2.0 a v jednej miestnosti ich môže byť až šestnásť. V prípade Lighsouse 1.0 to nebolo možné práve kvôli emitorom synchronizačných pulzov. Špecifikácie samotného headsetu Valve Index sú nasledovné:

- dva LCD displeje s rozlíšením 1440x1600
- obnovovaciu frekvenciu až 120 Hz s možnosťou aktivovania expermintálneho módu, ktorý umožňuje obnovovaciu frekvenciu až 144 Hz
- šírka zorného poľa 130 stupňov

Zapojenie je podobné ako u HTC Vive, USB 3.0 a klasický DisplayPort. Kvôli vyššej obnovovacej frekvencií ako u predošlých headsetoch sa vyslovene neodporúča používať HDMI na zapojenie headsetu.

Najväčšou inováciou sa však zaručene stal redizajnovaný ovládač. Ovládače Valve Index, počas vývoja označované ako „Knuckles“, sú spolu z headsetom a majákmi Lighthiuse 2.0 na obrázku 2.13. Knuckles ovládače obsahujú až 87 senzorov a ich dizajn, je navrhnutý tak, aby boli schopné sledovať nielen pozíciu a rotáciu samotného ovládača, ale zároveň aj prstov. Príklad ako daný ovládač funguje v realite a jeho odzrkadlenie vo VR je na obrázku 2.14.

Obr. 2.13: Headset Valve Index, senzory Lighthouse 2.0, ovládače Valve Knuckles.

Obr. 2.14: Valve Knuckles a ich sledovanie prstov, screenshot z videa [3]

.

2.6 Nevoľnosť z pohybu vo VR

S príchodom virtuálnej reality prišlo veľa pozitív. Nepriniesla len nový spôsob ako zažiť hranie hier, umožňuje ľudom virtuálne cestovať na miesta, kam vždy chceli ísť, umožňuje trénovanie ľudí v profesiách, ktoré sú nebezpečné, môže byť využitá na návrh či už strojových komponentov alebo rôznych interiérov. Používanie VR je obecne prínosom pre spoločnosť, avšak nezaobíde sa bez chýb. Jedným z najznámejších problémov, ktorý sa vyskytuje práve pri používaní VR headsetov, je nevoľnosť z pohybu, odborne kinetóza. Jedná sa o fyzický stav jedinca, ktorý sa prejavuje závratami, vyčerpaním, prípadne zdvíhaním žalúdku. Tento stav nastáva ak náš mozog obdrží zmiešané signály o pohybe vo svojom prostredí, inými slovami, oči mu vravia že sa pohybujeme, ale naše telo je v pokoji. U väčšiny ľudí sú častými spúšťačmi doprava autom, vlakom, jazda na kolotoči. Virtuálna realita predstavuje teda závažný problém, keďže pri napojení na VR je naše fyzické telo veľmi často v pokoji alebo sa pohybuje iba vo veľmi obmedzenom priestore.

Kinetóza vo virtuálnej realite je ovplyvnená niekoľkými faktormi. Medzi základné, ktoré ovplyvňujú vznik kinetózy je typ headsetu, typ ovládača, odozva a samotný návrh aplikácie. Typ headsetu, typ ovládačov a odozva predstavujú najdôležitejšie faktory a najviac ovplyvňujú či daný užívateľ bude pociťovať kinetózu. Prvotné headsety umožňovali sledovanie iba:

- pohľadu doprava a doľava
- pohľadu hore a dole
- naklonenie doprava a doľava

Keďže tieto headsety umožňovali pohyb iba v týchto troch dimenziách označujeme ich ako 3-DoF headsety [16]. Skrata „DoF“ je z anglického „Degrees of Freedom“, čo v preklade znamená stupne slobody. Toto bol však značný problém, keďže headsety neumožňovali pohyb vo virtálnom prostredí, tak pohyb musel byť dosiahnutý iným zariadením, v prípade prvotných balení headsetu Oculus Rift to bol gamepad Xbox One od spoločnosti Mirosoft. Problém, ktorý táto kombinácia spôsobovala, sa do istej miery vyriešil a všetky vyššie spomínané headsety dnes už podporujú 6-DoF. Headsety s podporou 6-DoF v rámci virtuálneho prostredia umožňujú:

- pohľadu doprava a doľava
- pohľadu hore a dole
- naklonenie doprava a doľava
- pohyb doprava a doľava
- pohyb dopredu a dozadu
- pohyb hore a dole

[3]video dostupné na https://www.youtube.com/watch?v=cjXSXmHZP3Q&ab_channel=SadlyItsBradley

Odozva a frekvencia vykresľovania podobne prispievajú ku vzniku kinetózy. Ak je vykresľovanie priestoru vo virtuálnej realite pomalé alebo samotná odozva je pomalá dochádza k desynchronizácií pohybu hlavy a vykresľovania, čo vedie k problému.

Ďalším faktorom, ktorým do istej miery môžeme zmierniť príznaky kinetózy vo VR je samotný pohyb v aplikácii. Snažíme sa pohyb riešiť tak, aby sa zabránilo priamemu pohybu vo virtuálnom priestore pomocou ovládačov. Ak sa jedná o aplikácie, v ktorých je hráč nútený prekonávať väčšie vzdialenosti práve za pomoci ovládačov, je nutné riadiť sa dvomi prístupmi:

- využiť teleportáciu
- v prípade nutnosti rotácie hráča, rotovať o pevne daný uhol

Väčšina aplikácií v dnešnej dobe prioritizuje tieto dva prístupy a používajú ich ako predvolené nastavenia. Užívateľ má možnosť si prepnúť nastavenia na plynulý pohyb využívaný v klasických hrách, avšak na vlastné riziko. Odolnosť voči kinetóze má každý inú, avšak dá sa do istej miery budovať. Či už pri hraní hier alebo práci vo VR sa odporúčajú pravidelné pauzy a obecne sa spočiatku nezdržiavať veľmi dlho vo virtuálnom prostredí ale naopak krátke sedenia a postupne ich navyšovať.

Kapitola 3

Neeuklidovský priestor

V nasledujúcich podkapitolách sa čitateľ oboznámi s pojmom neeuklidovského priestoru a jeho matematickým popisom, zároveň bude jedna podkapitola dedikovaná ako takýto priestor do istej miery implementovať v počítačových hrách, aby to samotného hráča bavilo a fascinovalo. Hlavným zdrojom, ktorý popisuje neeuklidovské priestory sa stal článok od doktora Martina Skrodzkiho [10].

3.1 Euklidovská geometria, priestor a počiatky neeuklidovskej

Euklid z Alexandrie, alebo taktiež známy aj ako „otec geometrie“, bol grécky matematik, ktorý položil základy chápania troj-dimenzionálneho priestoru. Predpokladá sa, že bol narodený zhruba okolo roku 325 pred Kristom a jeho životným dielom, ktoré sa používa dodnes bolo stanovenie piatich postulátov geometrie, dnes nazývanej aj ako Euklidovskej geometrie alebo Euklidovského priestoru. Týchto päť postulátov, pravidiel znelo nasledovne:

1. Každé dva rôzne body môžeme spojiť vždy len jedinou priamkou.
2. Ohraničenú priamu čiaru možno nepretržite rovno predlžovať.
3. Z ľubovoľného stredu možno opísať kružnicu s ľubovoľným polomerom.
4. Všetky pravé uhly sa navzájom rovnajú (sú zhodné)
5. Nech sú dané dve priamky, ktoré sú preťaté treťou priamkou. Ak zvierajú s ňou po jednej jej strane vnútorné uhly, ktorých súčet je menší než dva pravé uhly, tak sa pretínajú na tej istej strane.

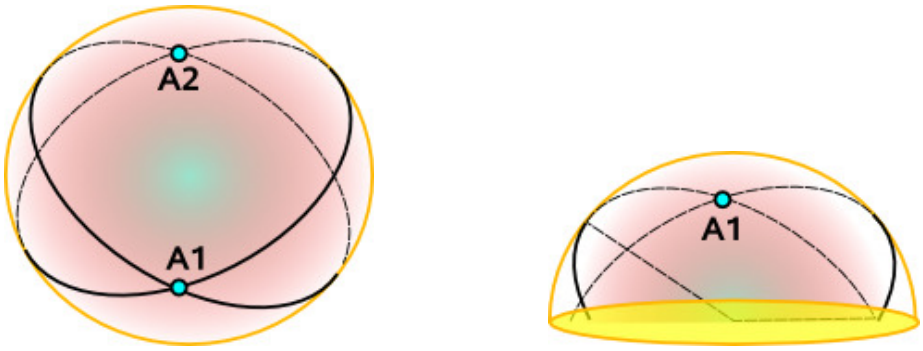
Posledný, piaty postulát, sa taktiež nazýva aj ako „axióm rovnobežných čiar“ a vďaka svojej krkolomnej formulácii sa dlhé roky rôzny matematici snažili dokázať, jeho závislosť na predošlých štyroch postulátoch. Tým by sa docielilo, že jednoducho by sa mohol daný postulát zrušiť, keďže by bolo možné ho odvodiť z predošlých štyroch. Prielom nastal až na konci 19. storočia, kde sa trom, na sebe nezávislým matematikom, podarilo dokázať presný opak. Prvou osobou, ktorý sa zmienil vo svojich dopisoch, že piaty Euklidovský postulát je nezávislý na predošlých, bol Carl Friedrich Gauss. Vo svojich dopisoch popisoval priestor, ktorý narušuje piaty Euklidov postulát. Toto však bolo v rozpore s filozofiou Immanueala Kanta o vnímaní priestoru. Gauss sa bál reakcie a kritiky nasledovateľov tejto filozofie a tak svoju prácu nikdy nezverejnil.

Zhruba o dvadsať rokov neskôr ruský matematik Nikolai Ivanovich Lobachevsky publikoval článok popisujúci neeuklidovskú geometriu. Tento článok bol neskôr zverejnený v miestnych novinách a následne preložený do ďalších jazykov. V rovnakom čase maďarský matematik János Bólyai vydal prílohu k učebnici matematiky, ktorú napísal jeho otec. V tejto prílohe jasne definoval priestor, ktorý podobne ako Gaussov a Lobachevskyho priestor, narušoval piaty Euklidov postulát. Aj keď jednotliví matematici pracovali úplne nezávisle tak Gaussove dopisy, Lobachevskyho články i Bólyaiova príloha, popisovali takmer identickú reprezentáciu neeuklidovskej geometrie (prípadne neeuklidovsky priestor), ktorú dnes poznáme pod názvom „hyperbolická geometria“.

Následne si matematici po celom svete začali uvedomovať, že zamenením piateho postulátu za iný, kľudne aj nimi navrhnutým postulátom, sú stále schopný produkovať zmysluplnú geometriu. V dnešnej dobe sú najznámejšími neeuklidovskými geometriami hyperbolická geometria (priestor), sférická geometria (priestor) a z nej odvodená eliptická geometria (priestor).

3.1.1 Sférická geometria

Zamenením piateho Euklidovského postulátu na pravidlo v znení: „Pre daný bod ležiaci na priamke, neexistuje žiadna rovnobežka“, dostávame geometriu, ktorú nazývame sférická geometria. Analógiou k tejto geometrii je samotná naša zemeguľa. Ako príklad si môžeme zvoliť severný a južný pól a poludníky, ktoré ich obe pretínajú. Vzdialenosti medzi jednotlivými poludníkmi sú najväčšie v okolí rovníka a postupne sa skracujú až dokým sa nepretnú v póloch. Sférickou geometriou rozumieme zaoblené priestory v ktorých sa všetky priamky pretínajú práve v dvoch bodoch. Tieto body sa nazývajú *antinodálne* a ležia na opačných stranách sféry. Eliptická geometria je špeciálny prípad sférickej, pre ktorú platí, že priamky sa pretínajú iba jeden krát. Z obrázka 3.1 je zrejmé, že daná geometria nespĺňa hneď viacero Euklidovsých postulátov.



Obr. 3.1: Príklad sférická geometrie s vyznačenými antinodálnymi bodmi na ľavo a príklad eliptickej geometrie na pravo [1].

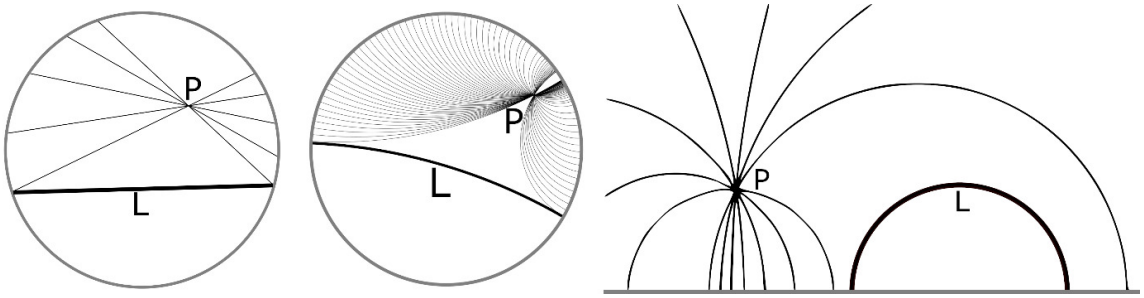
3.1.2 Hyperbolická geometria

Ďalším typom neeuklidovskej geometrie, je hyperbolická geometria. Práve o tomto type geometrie, priestore, sa vo svojich publikáciách zmieňovali vyššie spomínaní matematici.

[1] obrázok prebratý z https://www.euclideanspace.com/maths/geometry/space/nonEuclid/index.htm

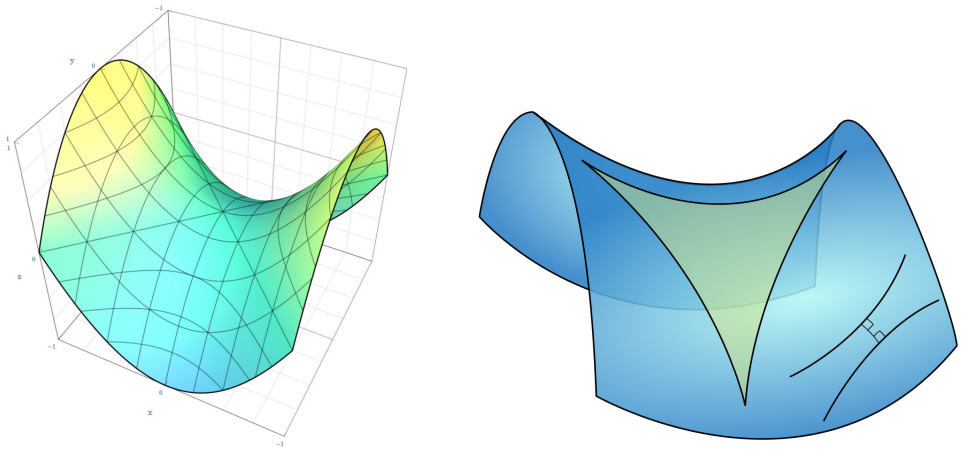
Tento typ geometrie vzniká podobným spôsobom ako sférická, zamenením piateho postulátu. Nový postulát znie: „Pre priamku P a bod I, neležiaci na P, kde P a I ležia na rovine J, existujú aspoň dve odlišné priamky, ktoré sa pretínajú v bode I ale nikdy nepretnú priamku P.“ Existuje niekoľko vizualizácií tohoto priestoru, všetky pomenované po známych matematikoch. Na obrázku 3.2 sú znázornené 3 typy vizualizácie:

1. v ľavo Beltrami-Kleinov model v disku, priamky sú rovné čiary
2. v strede Poincarého diskový model, priamky sú zaoblené
3. v pravo Poincarého model v polovičnej rovine, kde priamky sú buď rovné alebo zaoblené čiary ale uhly ostávajú nedotknuté



Obr. 3.2: Priamka L a zopár paralelných čiar ku nej prechádzajúcich bodom P v rôznych vizualizáciach hyperbolického priestoru.

Ďalšou známou ilustráciou tohoto priestoru v 3D je „konské sedlo“, obrázok 3.3.

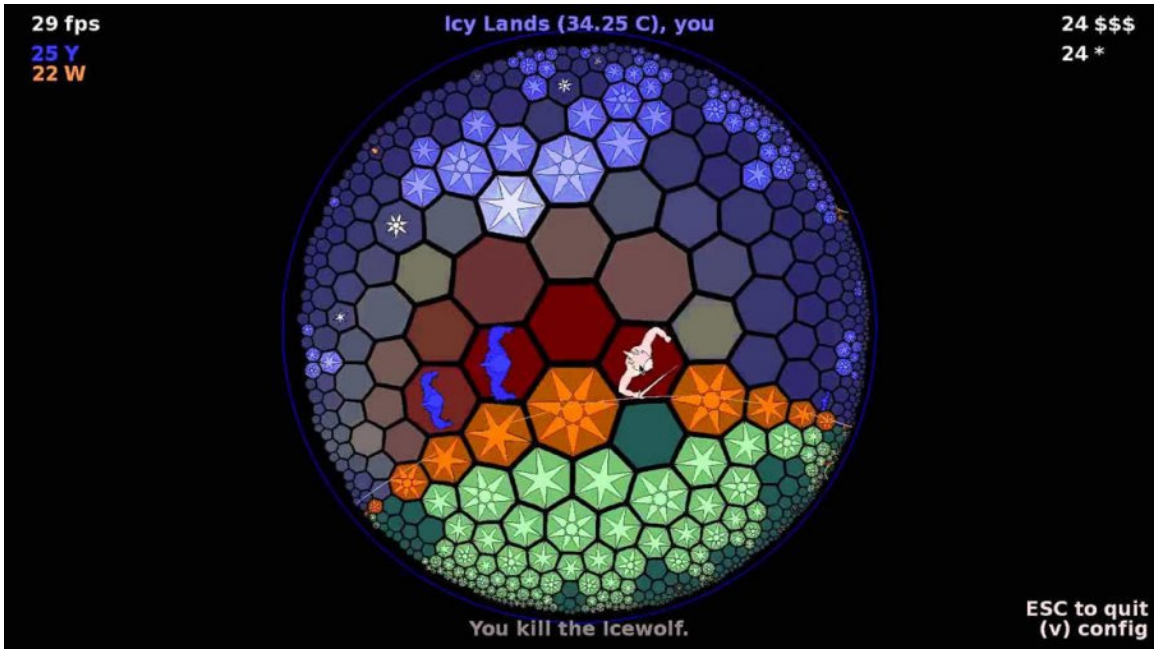


Obr. 3.3: Hyberbolické paraboloidy a v nich vyznačený trojuholník.

3.2 Neeuklidovský priestor v hrách

Experimety v oblasti vnímania priestoru sú čoraz viac zaujímavejšou témou aj pre herných vývojárov. Síce dnešné počítače už sú schopné vykreslovať matematicky presne definované neeuklidovské priestory, popísané v predošlej podkapitole, bohužiaľ veľké uplatnenie v hernom priemysle zatiaľ nenašli. Takéto priestory síce znejú na papieri veľmi zaujímavo,

neposkytujú však veľmi zaujímavé prostredie pre tvorbu hier. Príklady hier, ktoré sú tvorené presne na základe týchto definícií je možné vidieť na obrázkoch 3.4 a 3.5. Aj keď sa jedná o určitý vývojársky úspech, pre bežného hráča, ktorý nemusí mať potrebnú znalosť matematiky, to v prípade HyperRogue 3.4 môže pripadať ako bežná top-down roguelike hra. V prípade Hyperbolic-i, si takýto hráč môže povedať že sa jedná o klasickú adventúru s „divným“ zkreslením.



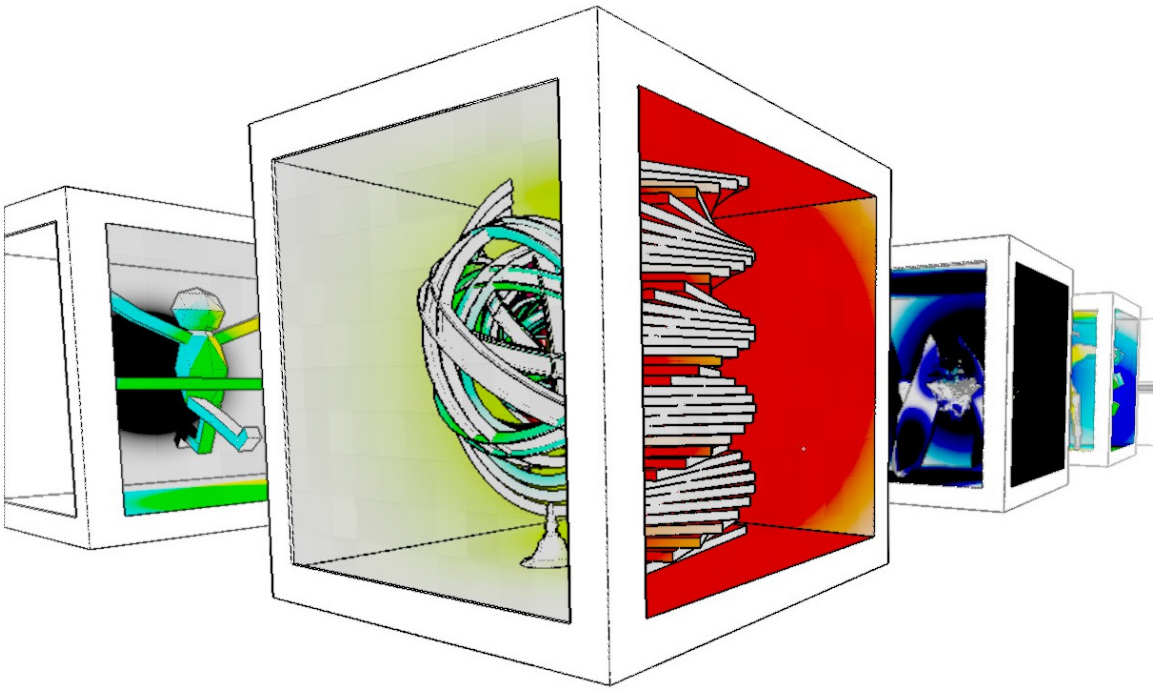
Obr. 3.4: Ukážka 2D hyperbolického priestoru z hry HyperRogue.

Obr. 3.5: Hyperbolica: v ľavo sférický priestor, v pravo hyperbolický.

Z tohoto dôvodu sa vo väčšine prípadov snažia vývojári navrhovať hry tak, aby spĺňali Euklidove postuláty a efekt neeuklidovského priestoru sa snažia docieliť inými technikami a metódami.

3.2.1 Portály

Jedným z prístupov, ktorý sa osvedčil v hernom priemysle na simuláciu neeuklidovských priestorov, sú portály. Jedná sa o neviditeľne steny, ktoré pri interakcií postavy hráča s nimi, prenesú samotnú postavu na iné miesto. Tento princíp bol do značnej miery spopularizovaný hrou Portal od spoločnosti Valve, v roku 2007 a následne jej pokračovaním Portal 2, v roku 2011. Samotné portály v týchto hrách však boli jasne viditeľné a v scéne, resp. v úrovni sa vždy nachádzali maximálne dva portály, vytvorené hráčom. Obidve hry boli Euklidovské, avšak princípi a možnosti, ktoré priniesli sa stali inšpiráciou pre ďalších vývojárov. Ako hru, ktorá využila práve tento potenciál portálov, si uvedieme „Antichamber“ (2013) od austrálskeho vývojára menom Alexander Bruce. Hra využívala neviditeľné portály a pomocou nich vykresľovala objekty, ktoré narušovali Euklidovské chápanie sveta. Samotná konštrukcia portálov je pomerne jednoduchá. Jedná sa o kus geometrie, obvykle jednoduchý štvorec, ktorý má na sebe nanesenú textúru, ktorá je vykreslená z pohľadu virtuálnej, neviditeľnej kamery niekde v scéne. Prípadne je možné využiť dodatočné buffery grafickej karty na maskovanie objektov v scéne. Príklad takéhoto efektu, priamo z hry Antichamber, je na obrázku 3.6. Je na ňom možné vidieť niekoľko kociek a v ich vnútri sa z každej strany nachádza iný objekt, respektíve, z každej strany kocky zaberá iný objekt rovnaké miesto v priestore. Oddelene sú jednotlivé objekty i kocka vykreslené podľa Euklidovskej definície priestoru. Dohromady však vzniká niečo, čo sa matematicky len veľmi ťažko popisuje.

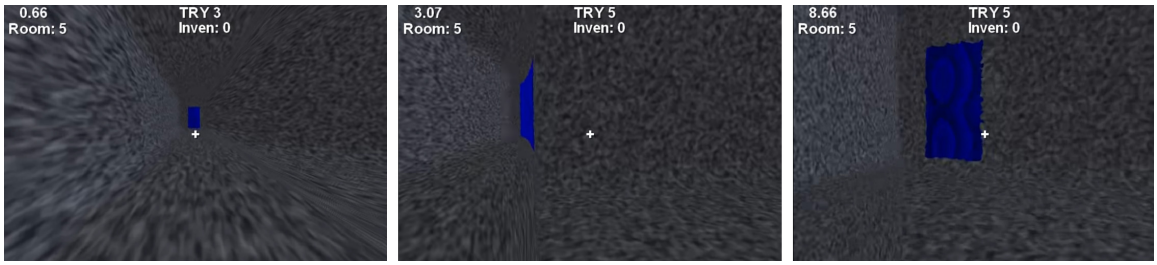


Obr. 3.6: Screenshot z hry Antichamber, v jednej kocke viacero objektov maskovaných stencilovým bufferom.

3.2.2 Ray-tracing

Jedným z ďalších riešení ako pristupovať ku vykresľovaniu neeuklidovského priestoru je ray-tracing. Pri využití tohoto prístupu sa vysielajú lúče z pozície hráča a na základe nich je okolitý priestor buď zmŕštený, roztiahnutý alebo dokonca aj istým spôsobom zaoblený.

Príkladom, ktorý využíva ray-tracing k vykresľovaniu neeuklidovského priestoru si zoberieme hru „No! Euclid!“. V príklade z obrázka 3.7, autor zostrojil miestnosť pozostávajúcu z jedného dlhého koridoru. Úlohou je dostať sa na druhú stranu za daný časový limit. Euklidovským rozmýšľaním, najkratšia cesta medzi dvomi bodmi je úsečka, je nemožné túto miestnosť prejsť. Hráč musí porozmýšlať prejsť doprava, kde sa mu samotná cesta skráti a až potom je možné miestnosť zdolať v časovom limite. Vlastnosti tohoto koridoru sú totiž upravované vystreľovanými lúčmi z kamery.



Obr. 3.7: NoEuclid!, naľavo dlhý koridor, v strede prechod v ktorom dochádza k zmršteniu priestoru, v pravo zmŕštený priestor.

Kapitola 4

Návrh aplikácie a implementácia

V predošlých kapitolách sme sa ponorili do technológie virtuálnej reality a terminológie neeuklidovského priestoru. Na tvorbu aplikácie ktorá mieša princípy a teórie spomínané vyššie bol použitý herný engine Unity. Nasledujúca časť textu oboznámi čitateľa so samotným enginom a základnými princípmi práce v ňom. Ďalšie kapitoly budú podrobne popisovať implementácie rôznych techník a metód na docielenie požadovaného neeuklidovského efektu, práve v tomto engine. Na popis enginu Unity bola použitá oficiálna dokumentácia [18], v prípade vykresľovania VR v Unity bol použitý článok [11] dotupný na stránkach Unity.

4.1 Unity Engine

Unity je moderný herný engine, napísaný v jazyku C++ a jeho hlavným cieľom je poskytnúť užívateľovi plne integrované prostredie na vývoj hier. Samotný engine je multi-platformný a aplikácie vyvíjané v ňom je možné spúšťat takmer na každom dnes dostupnom zariadení. Samotný editor na tvorbu hier je spustiteľný na desktopových zariadeniach:

- Widnows
- macOS
- distribúcie Linuxu

Zoznam najpopulárnejších platforiem na ktorých je možné spúšťat aplikácie vytvorené za pomoci Unity enginu je nasledovný:

- desktopové zariadenia:

  Widnows, macOS, distribúcie Linuxu

- mobilné zariadenia:

  Android, iOS, Tizen, UWP (Universal Windows Platform)

- konzolové systémy:

  od spoločnosty SONY: PlayStation 4-5, Vita

  od spoločnosty Microsoft: Xbox One

  od spoločnosty Nintendo: Switch, 3DS

- zariadenia umožňujúce virtuálnu realitu:

  Oculus Rift, GearVR (Samsung), SteamVR (HTC Vive, Valve Index)

Projekty vytvorené v Unity Engine majú špecifikú logickú štruktúru. Každý projekt pozostáva z jednej až viacerých scén. Jednotlivé scény reprezentujú nezávislé časti aplikácie, každá s vlastným nastavením. Môžu teda reprezentovať jednotlivé úrovne hry, prípadne nedostupné oblasti ku ktorým sa hráč dostane neskôr, kľudne môžu slúžiť aj na vykresľovanie inventára, prípadne rôznych in-game elementov ako hlavné menu. Scéna je tvorená hierarchickou stromovou štruktúrou herných objektov (anglicky „GameObject“). Herné objekty pozostávajú z jednej až viacerých komponentov, kde komponentom herného objektu, môže byť kľudne aj ďalší herný objekt. Herný objekt v Unity engine teda môžeme chápať ako kontajner, ktorý nadobúda zmysel až po vložení jednotlivých komponentov. Ďalšou vlastnosťou herného enginu Unity, ktorá uľahčuje prácu vývojárom, je možnosť ukladať si vytvorené a upravené herné objekty, prípadne i celé hierarchie herných objekov, samostatne do špeciálnych šablón nazvaných **Prefabrications**, v skratke **Prefabs**. Takto vytvorené šablóny je potom možné použiť v rámci ľubovolnej scény a nielen pred pustením ale aj dynamicky za behu aplikácie. Nasledovné zmeny tejto šablóny sa prejavia v každom hernom objekte, ktorý bol na základe nej vytvorený. Zároveň je možné nastaviť jednotlivým herným objektom rôzne označania (anglicky „Tag“) a vrstvy (anglicky „Layers“). Pomocou nich sme schopný jednotlivé objekty ľahšie vyhľadávať, manipulovať v rámci skriptov alebo úplne vymaskovať pri vykresľovaní.

4.2 Vykresľovacie reťazce enginu Unity

Na tvorbu hier v Unity engine je potrebný Unity Editor. Tento editor slúži na návrh vyššie spomínaných scén, ich editáciu a obecne zaobaluje funkcionalitu samotného enginu do užívateľsky príjemného a hlavne prístupného prostredia. Pri tvorbe nového projektu je nutné zadať názov daného projektu a zvoliť si vykresľovací reťazec. Keďže je Unity Engine komplexný herný engine poskytujúci širokú škálu rôznych grafických nastavení, post-processing efektov a iné, je nutné tento reťazec voliť vhodne na základe vyvíjanej aplikácie. Unity Engine vo verzií 2020.2 poskytuje celkovo štyri možnosti pri voľbe vykresľovacieho reťazca:

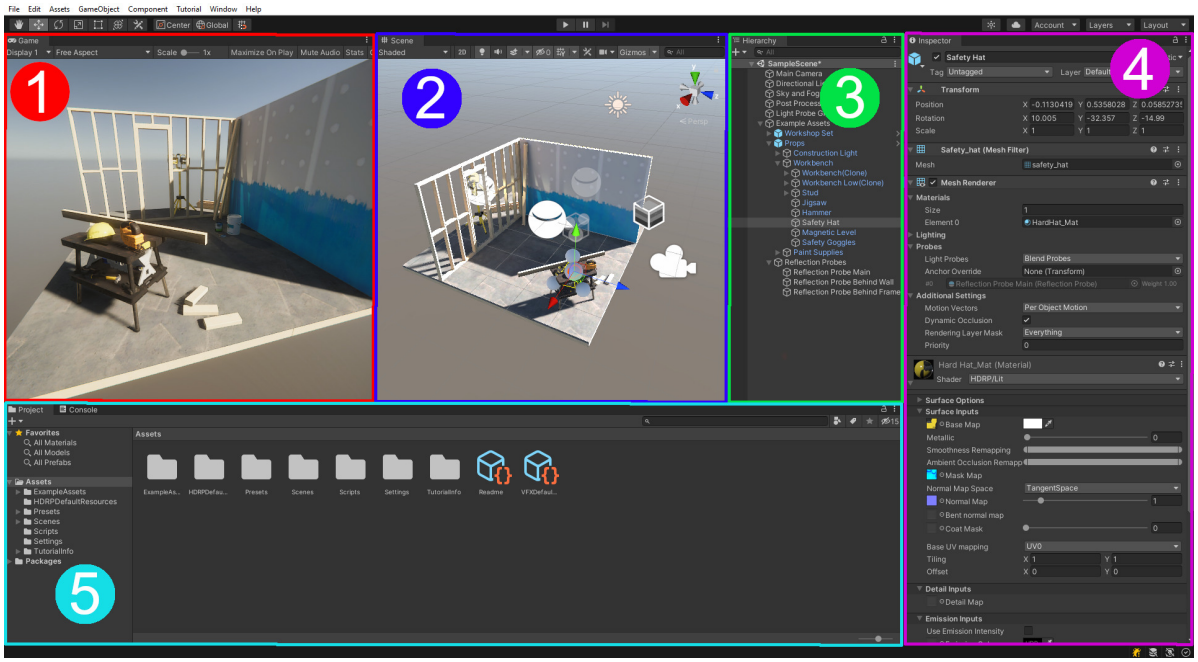
- vstavaný vykresľovací reťazec (anglicky „Built-in render pipeline“)
- univerzálny vykresľovací reťazec (anglicky „Universal Render Pipeline“, skratka URP)
- vykresľovací reťazec s vysokým rozlíšením (anglicky „High Definition Render Pipeline“, skratka HDRP)
- skriptovací vykresľovací reťazec (anglicky „Scriptable Render Pipeline“, skratka SRP)

Vstavaný vykresľovací reťazec je určený na všeobecné vykresľovanie. Je najprístupnejší avšak je zo všetkých najviac limitovaný. Univerzálny vykresľovací reťazec predstavuje vlastne šablónu z istými preddefinovanými nastaveniami skriptovacieho reťazca. Je vhodný na vývoj aplikácií ako na mobilné zariadenia tak aj na konzole, notebooky a stolové počítače. Podobne ako URP tak aj HDRP, teda vykresľovací reťazec s vysokým rozlíšením, je len šablóna z preddefinovanými nastaveniami skriptovacieho reťazca. Na rozdiel od univerzálneho vykresľovacieho reťazca, je HDRP zamerané primárne na grafiku. Umožňuje priame alebo odložené vykresľovanie (forward, deferred rendering) ako aj akceleráciu rady metód na grafických kartách s použitím compute shaderov.

Skriptovací vykresľovací reťazec je zo všetkých najversatilnejší, pretože ako to aj zo samotného názvu vyplýva, umožňuje programátorovi riadiť samotné vykresľovanie pomocou skriptov napísaných v jazyku C#. Po voľbe reťazca sa spustí samotný editor.

4.3 Rozhranie Unity editoru

Rozhranie editoru pozostáva z niekoľkých dôležitých častí. Tieto časti sú obzvlášť prispôsobené koncepcii scén na ktorej je Unity Engine založený. Časti rozdeľujeme na pohľady (anglicky „views“) a okná (anglicky „windows“). Pri klasickom rozložení sa užívateľovi po zapnutí a zvolení projektu naskytnú tri okná a dva poľady. Tieto pohľady, okná, je možné pridávať, odoberať prípadne meniť samotné ich rozloženie. Klasické rozloženie editoru, je na obrázku 4.1.



Obr. 4.1: Unity Editor s vyznačenými hlavnými pohľadmi. vyznačené časti predstavujú: 1 - *Game*: pohľad na spustenú hru, 2 - *Scene*: pohľad na scénu, 3 - *Hierarchy*: okno na hierarchiu herných objektov, 4 - *Inspector*: okno na aktuálne zvolený prvok, 5 - *Project*: okno so zdrojmi projektu.

4.3.1 Pohľad na spustenú hru

V tomto prípade sa jedná o pohľad ktorý nám zobrazuje vytvorenú scénu s pohľadu hlavnej kamery. V scéne môže byť obecne niekoľko kamier. Počas behu aplikácie môže byť ako hlavná označená práve jedna kamera. Hlavná kamera slúži na zobrazenie sveta pre hráča a jej prítomnosť v scéne je nevyhnutná. Čo daná kamera vidí zobrazuje práve tento pohľad v skutočnom čase.

4.3.2 Pohľad na scénu

Pohľad na scénu reprezentuje aktuálne zvolenú scénu a vizualizuje obsah v nej. Je prispôsobený na jednoduchú a prehľadnú manipuláciou s aktuálne zvolenou scénou. Za pomoci

pohľadu na scénu sme schopný manipulovať s objektami, ktoré sa v nej nachádzajú. Medzi základne operácie v rámci tohoto pohľadu patrí možnosť vybrať, premiestniť, upraviť rotáciu objektu prípadne jeho mierku. Okrem manipulácie s objektami, taktiež umožňuje prepínanie medzi 2D a 3D zobrazením, v 3D umožňuje prepínanie medzi ortogonálnou a ortografickou kamerou, umožňuje prepínanie medzi vykresľovaním objektov s naneseným materiálom alebo len ich drôteným rámom (anglicky „wireframe“). Taktiež umožňuje jednoduchým kliknutím na príslušné tlačidlá vypnúť/ zapnúť vykresľovanie skyboxu, hmly prípadne aj časticových systémov.

4.3.3 Okno na hierarchiu projektu

Jedná sa o okno, ktoré obsahuje herné objekty a ich hierarchické zaradenie v aktuálne zvolenej scéne. Umožňuje danú hierarchiu upravovať podľa potrieb pridávaním nových herných objektov, odoberaním a ich presúvaním v hierarchii. Herné objekty sú v rámci tohoto pohľadu reprezentované ako uzly. Koreňový uzol predstavuje samotná scéna. Uzly, respektíve herné objekty je možné vnárať do seba, tak ako to bolo popísane v úvode kapitoly. Po zvolení uzlu (herného objektu) v tomto pohľade sa zobrazia v okne *Inspector* všetky jeho komponenty, rovnako ako sa v pohľade *Scene* označia všetky herné objekty, ktoré takto zvolený uzol v hierarchií obsahuje. Po zvolení niektorého nadradeného uzlu a následne upravením vlastností (viď. kapitola 4.3.4), sa zmenia aj vlastnosti podradených uzlov.

4.3.4 Okno s aktuálne zvoleným prvkom

Okno, *Inspector*, zobrazuje svoj obsah dynamicky, na základe zvoleného herného objektu v scéne. Zobrazuje jednotlivé komponenty aktuálne zvoleného herného objektu. Zvoliť herný objekt môžeme pomocou vyššie spomínaného okna na hierarchiu alebo jednoduchým výberom v pohľade na scénu. V tomto okne môžeme upravovať jednotlivé komponenty a ich vlastnosti. Taktiež môžeme dané komponenty zvolenému hernému objektu pridávať alebo odoberať.

4.3.5 Okno so zdrojmi projektu

V nie poslednom rade si popíšeme okno so zdrojmi projektu. V tomto okne je zobrazená samotná štruktúra zdrojov aktuálne otvoreného projektu. Medzi zdroje projektu spadá všetko čo v danom projekte istým spôsobom využívame. Patria sem jednotlivé modely, materiály, textúry, shader programy, skripty, vytvorené prefabrikácie a kľudne aj samotné scény. Implicitne sú zobrazené len tie zdroje, ktoré sa nachádzajú v priečinku „*Assets*“. V ľavej časti tohoto okna z obrázka 4.1 sa nachádza panel reprezentujúci adresárovú štruktúru. Po zvolení konkrétneho adresára sa jeho obsah zobrazí v pravej časti. Samotnú adresárov=u štruktúru je možné priamo meniť v editore, prípadne si pridávať aj odkazy na priečinky so zdrojmi na rýchlejšiu navigáciu.

4.4 Virtuálna realite v Unity

Ako už bolo spomínané v úvode kapitoly, Unity Engine je multiplatformný a umožňuje nám vývoj aplikácií aj pre virtuálnu realitu. Obecne Unity umožňuje vývoj XR aplikácií, do ktorých virtuálna realita spadá ako podmnožina. Pod termýnom XR rozumieme:

- virtuálnu realitu (VR)

- miešanú realitu (MR)
- rozšírenú reality (AR)

Vývojový tým Unity je v blízkej spolupráci z rôznymi partnermi, ktorý poskytujú informácie o akých podporách by Unity malo byť rozšírené.

Na tvorbu aplikácie vo virtuálnej realite je potrebné si nainštalovať potrebné pluginy. Unity pomocou svojích pluginov, ktoré sú neustále vo vývoji, **XR Plug-in Managment** a **XR Interaction Tookit** dáva priamu podporu vyvíjať aplikácie vo VR. Na spracovanie vstupu, sledovanie zariadení slúži **XR Interaction Toolkit**, kde **XR Plug-in Management**, služi na správu jednotlivých zariadení. Pomocou **XR Plug-in Management** sme schopný stiahnuť si základné potrebné balíky pre vývoj na zariadenia a taktiež poskytuje možnosti prepínaia medzi samotným vykresľovaním do headsetu, viz kapitola 4.4.1. Na zariadenia Oculus postačia práve tieto pluginy, avšak existuje aj špecializovaný balík **Oculus SDK** od firmy Oculus, ktorý obsahuje rôzne príklady, či už spracovania vstupu alebo pohybu po scéne. Vývoj aplikácií na iné zariadenia, je možné pomocou SDK tretích strán. Pomocou **Google VR SDK** sme schopný vyvíjať aplikáciu na Google Daydream a Google Cardboard. Ďalším pluginom umožnujúcim tvorbu VR aplikácií je plugin **OpenVR**, už od známej firmy Valve (nemýliť si z **OpenXR**, vyvíjaný skupinou Khronos). OpenVR Unity plugin obsahuje potrebné knižnice na tvorbu a kompiláciu primárne na zariadenia SteamVR, teda headsety HTC Vive(a jeho potomkovia) a Valve Index. Balík by však mal obsahovať aj podporu pre Oculus a Windows Mixed Reality.

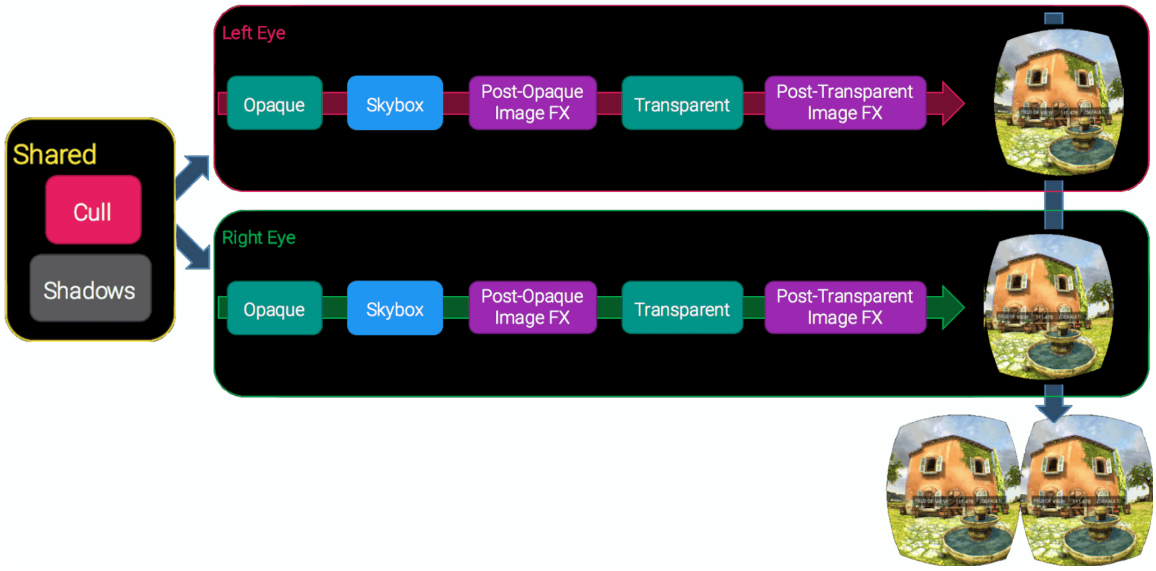
4.4.1 Vykresľovanie VR aplikácií

Ako už bolo v úvode kapitoly spomenuté voľba vhodného vykresľovacieho reťazca je jednou z najdôležitejších častí pri začiatkoch vývoji ľubovolného projektu. Jeho výber nebude ovplyvňovať nielen kvalitu grafických efektov ale aj výkon samotnej aplikácie na rôznych zariadeniach. Na vývoj aplikácií do VR sa obecne odporúča voľba univerzálneho vykresľovacieho reťazca URP prípadne reťazca s vysokým rozlíšením HDRP.

Ďalším problémom, ktorý však nastáva pri vykresľovaní aplikácií do virtuálnej reality je spôsob akým samotná scéna bude vykreslená. Na základe obecného headsetu sú potom k dispozícií tri rôzne spôsoby ako danú scénu vykresľovať a to:

- Multi-Pass rendering
- Single-Pass Stereo rendereing
- Single-Pass Instanced rendering

Viacpriechodové vykresľovanie alebo anglicky „Multi-Pass rendering“, je základné a najjednoduchšie vykresľovanie, s ktorým sa stretneme pri virtuálnej reality. Ako to aj z názvu vyplýva, jedná sa o vykresľovanie vo viacerých priechodoch. Scéna je vykreslená minimálne dvakrát, najprv z pohľadu ľavého oka a následne z pohľadu pravého oka. Grafický reťazec tohoto vykresľovania je znázornený na obrázku 4.2. Táto metóda vykresľovania je podporovaná všetkými spomenutými headsetami avšak oproti ostatným značne náročnejšia ako po procesorovej stránke, tak aj po grafickej.



Obr. 4.2: Multi-Pass rendering. Ako je možné vidieť, tak samotné orezávanie scény a výpočet tieňov je riešený jednotne pre obe oči. Dôvodom je snaha o optimalizáciu tohoto vykresľovania, výberom prvkov, metód, ktoré nie sú závislé na samotnej kamere. V prípade tieňov, je ich implementácia rozdelená do dvoch krokov: generovaním kaskádovej tieňovacej mapy (anglicky „cascaded shadow maps“) a mapovanie týchto tieňov do obrazovkového priestoru (anglicky „screen-space“). Samotné generovanie kaskádovej mapy je úplne nezávislé na pozícii a rotácii kamery, mapovanie do obrazovkového priestoru je však závisle na kamere a je nutné ho robiť v každom priechode reťazca zvlášť.

Ďalším spôsobom je vykresľovať jedným priechodom (anglicky „Single-Pass Stereo rendereing“). Namiesto vykresľovania scény pre každé oko zvlášť, je scéna vykreslená iba raz do jednej texúry. Táto textúra má však dvojnásobnú šírku ako textúra použitá v Multi-Pass vykresľovaní. Do prvej polovice sa vykreslí pohľad z ľavého oka a do druhej sa vykreslí pohľad z pravého oka. Tento spôsob je značne efektívnejší ako vykresľovanie Multi-Pass pretože sa jednotlivé objekty v rámci scény prechádzajú iba raz a vykresľuje sa len na základe prepínania **viewportov**[1]. Vylepšená verzia, ktorá znižuje počet vykresľovacích príkazov je vykresľovanie jedným priechodom s instancovaním.

4.5 Skriptovanie v Unity

Skripty v Unity Engine predstavujú komponenty herných objektov, pomocou ktorých sme schopný jednotlivým herným objektom pridávať novú funkcionalitu, prípadne upravovať starú. Sú neoddeliteľnou súčasťou vývoja a aj keď je možné teoreticky vytvoriť funkčnú aplikáciu už z predom definovaných skriptov a herných objektov, tak tvorba vlastných skriptov jednoznačne uľahčuje správu a prehľad aplikácie a do istej miery zvyšuje komplexnosť aplikácie a reprezentuje skúsenosti vývojára.

Na tvorbu daných skriptov slúži v Unity Engine jazyk C#. Písanie skriptov je možné v ľubovolnom textovom editore, prípadne i komplexnejších editoroch ako Microsoft Visual Studio alebo Visual Studio Code. Tieto editory je však nutné nainštalovať zvlášť a nastaviť

[1] oblasť zobrazujúca pohľad kamery

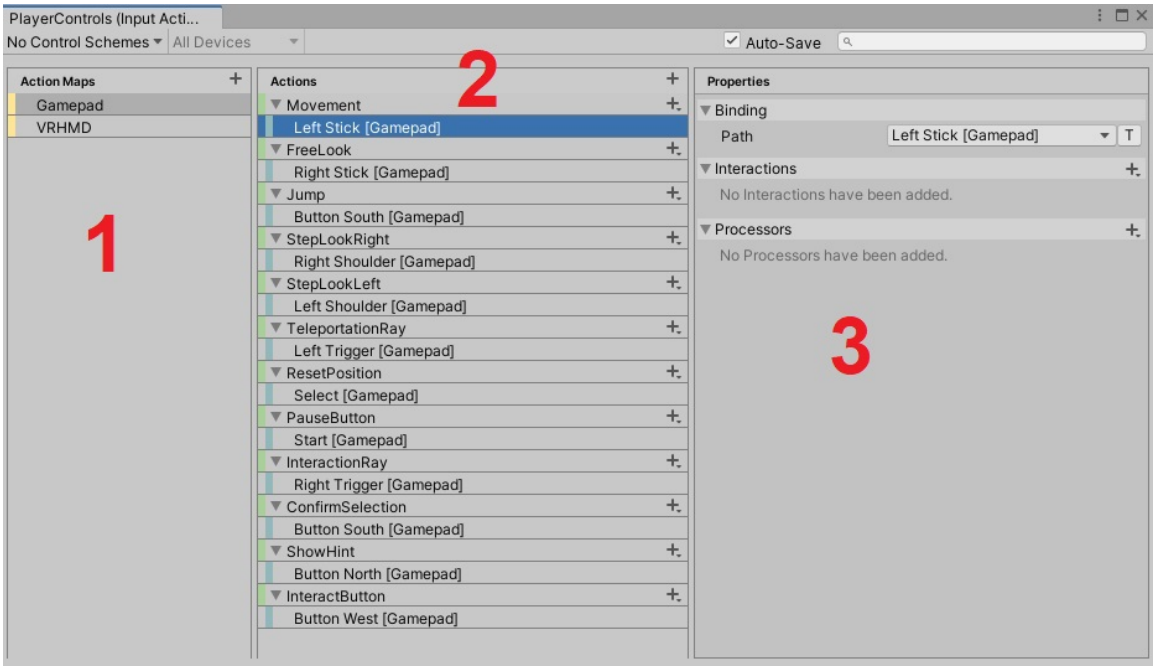
potrebné referencie na ne v Unity Editore. Staršie verzie Unity obsahovali MonoDevelop[2], jeho podpora však skončila v roku 2018. Výhodou využitia vyššie spomenutých editorov, je možnosť prepojenia z Unity, čo umožňuje jednoduché odladovanie týchto skriptov. Odladovanie prebieha potom podobne ako pri vývoji samostatnej aplikácie v daných editoroch. napríklad pridaním potrebných **breakpointov**, nastavením **watch** okien na sledovanie potrebných variabilných hodnôt a postupným krokovaním a prechádzaním tak zdrojového kódu.

Pri písaní vlastných skriptov v jazyku C# je nutné dodržiavať zopár jednoduchých pravidiel. V prípade, že daný skript má predstavovať komponentu herného objektu musí istým spôsobom dediť triedu **MonoBehaviour**. Keďže C# umožňuje dedičnosť stačí aby jedna bázová trieda dedila **MonoBehaviour**, od nej vytvorený potomkovia už nemusia. Táto trieda reprezentuje životný cyklus skriptu. Všetky triedy, ktoré ju dedia potom môžu upravovať istým spôsobom rôzne svoje vlastnosti v rôznych etapách životného cyklu. Implementovaním metódy **Start()** vo vlastnom skripte, zabezpečíme že sa táto funkcia vykoná práve raz, pri spúšťaní skriptu. Implementovaním metódy **Update()**, zabezpečíme, že sa daná funkcia bude vykonávať každý jeden snímok. Implementovaním metódy **Start()** môžeme docieliť teda prvotnú inicializáciu premenných a hodnôt a implementovaním metódy **Update()** môžeme docieliť, že sa v každom snímku bude počítať nová hodnota z predošlých. Príklad využitia môže byť jednoduchý pohyb hráča v scéne, kde je nutné počítať o koľko metrov sa hráč pohol a ktorým smerom. Ďalšími dôležitými metódami, bez ktorých by sa nezaobišiel takmer žiadny projekt sú **OnTriggerEnter**(...) a **OnTriggerExit**(...). Ich spracovanie prebieha vo fázy, kde sa vyhodnocuje fyzika enginu a slúžia na detekciu kolízií, respektíve vstup alebo výstup medzi objektami s komponentami **Collider**, **Character Controller**, **Rigidbody** a inými.

4.6 Postava hráča a pohyb hráča po scéne

Ako prvú vec bolo nutné implementovať jednoduchú postavu reprezentujúcu samotného hráča. Do scény bol vložený herný objekt, jeho **Tag** nastavený na „Player“ a ako potomok mu bola priradená hlavná kamera. Po jej priradený bolo nutné konvertovať hlavnú kameru na VR kameru. Umožňuje to vyššie spomínaný **XR-Interaction Toolkit**. Hlavná kamera je v scéne sledovaná pomocou komponenty **Tracked Pose Driver**, kde je nastavené sledovanie pozície a rotácie centrálneho oka headsetu. Na sledovanie samotných očí boli ako potomkovia hlavnej kamery vytvorené dva herné objekty, každý s rovnakou komponentou **Tracked Pose Driver**, s nastavením sledovania ako pre ľavé oko, tak aj pre pravé. Hernému objektu s tag-om „Player“ bola priradená komponenta **Character Controller**. Komponenta je zameraná na uľahčenie pohybu hráča po scéne a zároveň umožňuje aj detekciu kolízií a spúštanie **OnTriggerEnter**(...) a **OnTriggerExit**(...) akcií. Samotný **Character Controller** však nespracováva žiadne vstupy, či už z klávesnice alebo ovládačov. Na spracovanie vstupov bolo potrebné využiť balík **Input System** vo verzií 1.0.2 [17]. Tento systém umožňuje vytvoriť pre rôzne zariadenia jednotlivé mapovania a definovať aj hodnoty aké sa majú získavať. Obrázok 4.3 obsahuje vytvorené mapovania pre ovládač ako aj pre VR headset.

[2]open-source vývojové prostredie



Obr. 4.3: Vytvorené mapovanie pomocou balíka **InputSystem**. Jednotlivé položky v okne znamenajú: 1 - Action Maps, predstavuje kolekciu akcií. 2 - Actions, predstavuje akcie ktoré informujú kód o tom, že nastal daný vstup podľa vytvorenej väzby (anglicky „binding“). Vyznačená akcia „Movement“, je naviazaná na ľavý joystick ovládača. 3 - Properties, predstavuje okno s vlastnosťami zvolenej väzby.

Samotnú implementáciu, čo sa pri danej akcii má vykonať obsahuje skript **Player.cs**. Tu je v metóde **Awake()** vytvorený nový objekt **PlayerControls**. Tento objekt predstavuje vyššie spomínané mapovanie tlačidiel na akcie a následne sú priradené akciám funkcie, ktoré sa majú vykonať. Príklad spracovania vstupu je vo výpise 4.1.

```
1  void Awake(){
2  controls = new PlayerControls();
3  controls.Gamepad.Movement.performed += ContextMenu => Move(ContextMenu);
4  controls.Gamepad.Movement.canceled += ContextMenu => StopMove();
5  ...
6
7
8  ...
9
10 void Move(InputAction.CallbackContext ContextMenu) {
11    movement = ContextMenu.ReadValue<Vector2>();
12 }
```

Výpis 4.1: Príklad vytvorenia objektu **PlayerControls** v metóde **Awake()** a naviazanie akcie „Movement“ na funkcie podľa aktuálneho stavu akcie. Funkcia **Move(...)** číta dvojkový vector.

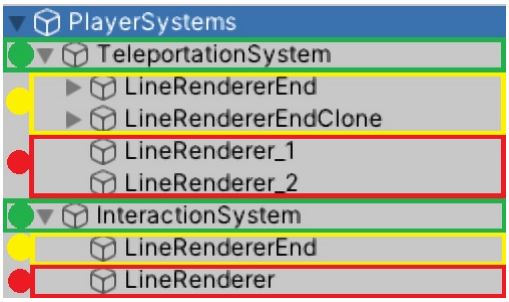
V momente, keď sa vykoná vstup z ľubovolného zariadenia ktoré je priradené k akcii **Movement** začne sa vykonávať priradená funkcia **Move(...)**. V kontexte tohoto projektu

táto funkcia získa iba horizontálnu a vertikálnu súradnicu ľavého joysticku na ovládači Xbox. V momente, keď sa prestane získavať vstup je vykonaná funkcia **StopMove()**. Pri získavaní hodnôt je postava hráča priebežne posúvaná pomocou metódy **CharacterController.Move(...)**.

Ako to už bolo spomenuté aj v kapitole 2.6, je potrebné dať možnosť hráčovi vyhnúť sa plynulému pohybu po scéne. Jednak pridaním rotovania postavy hráča o pevne daný uhol a možnosti sa voľne premiestňovať po scéne za pomoci teleportácie. V prípade pevne danej rotácie stáčí vytvoriť väzby na potrebné tlačidlá ovládača a postavu hráča otočiť o presne stanovený uhol. V prípade teleportačného lúču bol vytvorený a implementovaný systém popísaný v podkapitole 4.6.1

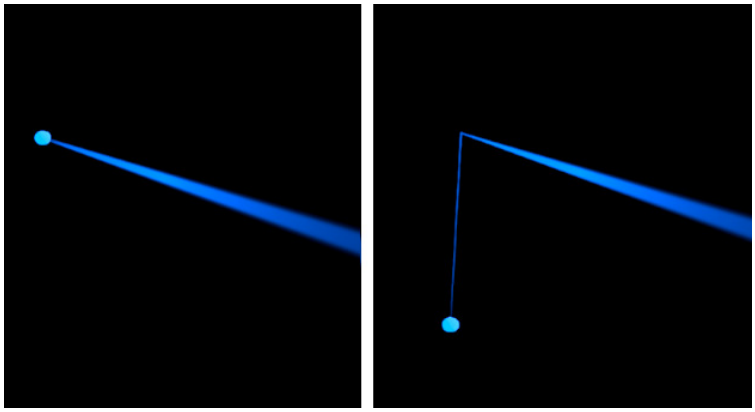
4.6.1 Teleportačný a interakčný systém

Teleportačný ako aj interakčný systém pozostáva z troch hlavných herných objektov, viz obrázok 4.4.



Obr. 4.4: Herné objekty teleportačného a interakčného systému v pohľadu na hierarchiu scény. Zelenou vyznečené objekty predstavujú rodičovské objekty. Žltou vyznačené objekty reprezentujú koniec ľúča, ktorý sa vykresľuje pri zásahu. Červenou vyznačené objekty reprezentujú samotný lúč.

V oboch prípadoch je základným stavebným blokom herný objekt s názvom „LineRenderer“. Tento objekt má ako komponentu skript **CustomLineRenderer.cs** a je zodpovedný za vystreľovanie segmentovaných lúčov do scény a detekciu trafeného objektu. Za pomoci tohoto skriptu je možné nastaviť aký dlhý bude jeden segment lúča, prípadne ak bude vystrelený lúč pozostávať z viacerých segmentov, tak aký bude medzi segmentami spád, obrázok 4.5. Počet z koľkých segmentov daný lúč bude pozostávať je pevne daný v nadradenom hernom objekte.



Obr. 4.5: Vystrelený jeden segment lúča a vystrelené 2 segmenty s nastavením veľmi ostrého spádu.

Segmentácia má v tomto prípade dva významy. Prvý význam nadobúda v momente ak daný lúč zasiahne objekt označený vrstvou „PortalCollider“. V tomto momente vieme, že lúč zasiahol portál a bude nutné vystreliť dodatočný, sekundárny, lúč. Na základe indexu segmentu, ktorý trafil portál, sme schopný dopočítať koľko segmentov môžeme preskočiť pri vykresľovaní sekundárneho lúča. Jeho druhý význam spočíva v tom, že nastavením rôznej dĺžky segmentu a spádu medzi nimi sme schopný napodobiť teleportačné lúče implementované v rôznych iných aplikáciách ako napríklad SteamVR, viz obrázok 4.6.

Obr. 4.6: Teleportačný lúč v aplikácii SteamVR[3]. Svetlo-zelená prerušovaná čiara reprezentuje teleportačný lúč. Tmavo-zelené štvorce na podlahe reprezentujú vyhradené miesto, kam sa hráč môže teleportovať.

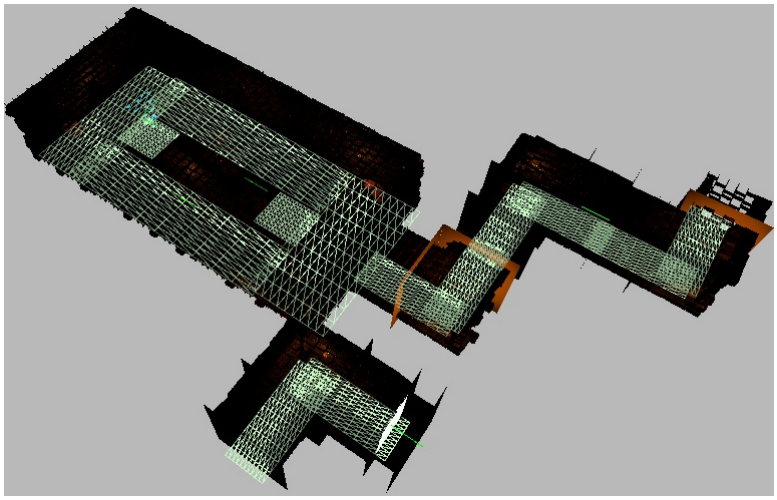
Na vykreslenie vystreleného lúča slúži komponenta **LineRenderer**, kdežto na detekciu zásahu slúži **PhysicsRaycast**. U teleportačného lúča sa detekujú objekty ktoré majú nastavené nasledovné vrstvy (Layer):

- Default
- PlayerPath
- PortalCollider
- TeleportNullifier
- Hint

V prípade že došlo k zásahu jednej z daných vrstiev sa samotné informácie o zasiahnutom objekte predajú nadradenému hernému objekte (vyznačenou zelenou farbou na obrázku 4.4) a tu sú potom spracované.

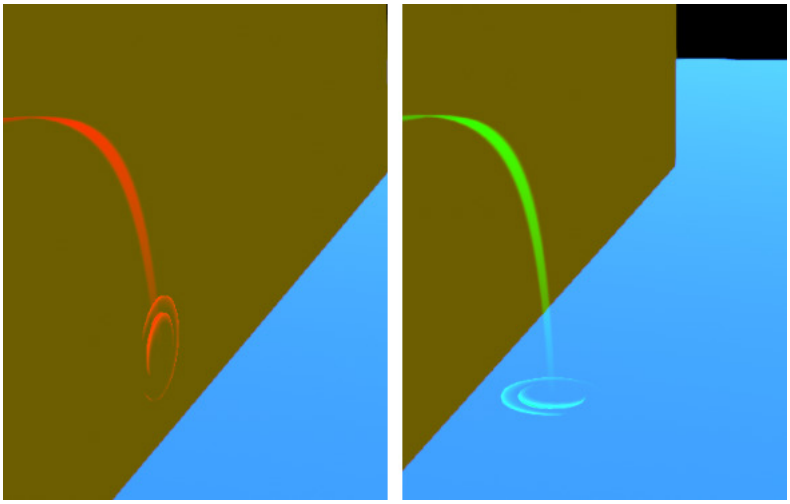
Vrstva „Default“ predstavuje takmer všetku geometriu v scéne. Pri zásahu objektu s danou vrstvou sa získa zasiahnutá pozícia a normála trojuholníka. Na túto pozíciu sa nastaví koncový bod (**LineRendererEnd**) telportačného alebo interakčného systému. Zásah je však vyhodnotený ako neuspešný, nedôjde k presunu hráča. Podobne je tomu tak aj v momente zásahu vrstvy označenej ako „TeleportNullifier“, vystreľovanie segmentov lúča však nekončí, ale poktračuje dokým sa nenarazí na ľubovolnú inú vrstvu.

Objekty s vrstvou „PlayerPath“, ako to aj z názvu vyplýva, reprezentujú vyhradenú cestu pre hráča. Vyhodnotenie pri zasiahnutí takýchto objektov je úspešné a hráč bude po uvoľnení tlačidla, ktorým sa striela lúč, prenesený na danú pozíciu. Výhodou využitia **PhysicsRaycast** je, že ak dôjde k zásahu, vráti sa svetová (anglicky „world-space“) pozícia zasiahnutého bodu, to znamená, že nie je nutný žiadny ďalší prepočet ani transformácia do lokálneho priestoru a bod sa môže priamo nastaviť ako nová pozícia hráča. Vrstva „Hint“ predstavuje, že pri zásahu sa hráčovi zobrazí menu s pomocou. Na obrázku 4.7 je úroveň s vyznačenou cestou pre hráča a na obrázku 4.8 zásah vrstvy „Default“ a vrstvy „PlayerPath“.



Obr. 4.7: Úroven sedem. Zelenou vyznačené objekty predstavujú vyhradenú cestu pre hráča. Pri ich zásahu dôjde k presunu hráča na novú pozíciu.

[3]obrázok prebratý zo stránky http://s824284416.websitehome.co.uk/blog/interaction-patterns/virtual-reality-patterns/teleport-lock-target/



Obr. 4.8: Teleportačný lúč. Podľa vyhodnotenia zásahu sa prepína medzi dvoma materiálmi. Na ľavo zasiahnutý objekt s vrstvou „Default“, ktorý zabraňuje teleportácii hráča a na pravo zasiahnutý objekt s vrstvou „PlayerPath“.

Najzaujímavejší prípad predstavujú objekty označené ako vrstva „PortalCollider“. Jediné objekty v scénach, ktoré majú nastavenú túto vrstvu sú portály, viz podkapitolu 4.7.5, a to konkrétne iba ich **BoxCollider**. Pri zásahu objektu je zrejmé ze nová pozicia hráča musí byť na opačnej strane portálu. Je teda nutné vystreliť sekundárny teleportačný lúč, práve z pozície prepojeného portálu k portálu zasiahnutého. V momente zásahu sú k dispozícii nasledovné informácie:

- HitPoint - bod zásahu **BoxCollideru** vo svetovej pozícii
- LinkedPortal - portál, s ktorým je zasiahnutý portál prepojený

Keďže samotný „HitPoint“ je uvedený v súradniciach celého sveta (anglicky „world-space coordinates“) je nutné v tomto prípade previesť transformáciu do lokálneho priestoru zasiahnutého **BoxCollideru**. Na transformáciu poslúži metóda **InverseTransformPoint**(Vector3 position). Funkcia sa volá vždy z transformu objektu ku ktorému chceme zistiť lokálnu pozíciu bodu. Pseudokód na získanie lokálneho zásahu na **Collidery** je nasledovný:

```
1    localInHitCollider = HitObject.transform.InverseTransformPoint(
        HitPoint);
```

Obdobným spôsobom získame world-space pozíciu s prepojeným portálom s využitím opačnej metódy **TransformPoint**(Vector3 position).

```
1    worldInLinkedCollider = LinkedCollider.transform.TransformPoint(
        localInHitCollider);
```

Sekundárny lúč sa bude vystrelovať z pozície **worldInLinkedCollider**. Smer sekundárneho lúča je určený vektorom **forward** kamery, ktorá je súčasťou portálu.

4.7 Neeuklidovské efekty

Tak, ako to už bolo nastienené v teoretickom úvode, na dosiahnutie neeuklidovských efektov alebo priestorov je možné využiť niekoľko rôznych metód a postupov. V tejto práci na dosiahnutia žiadaných efektov boli implementované dve základné metódy a to:

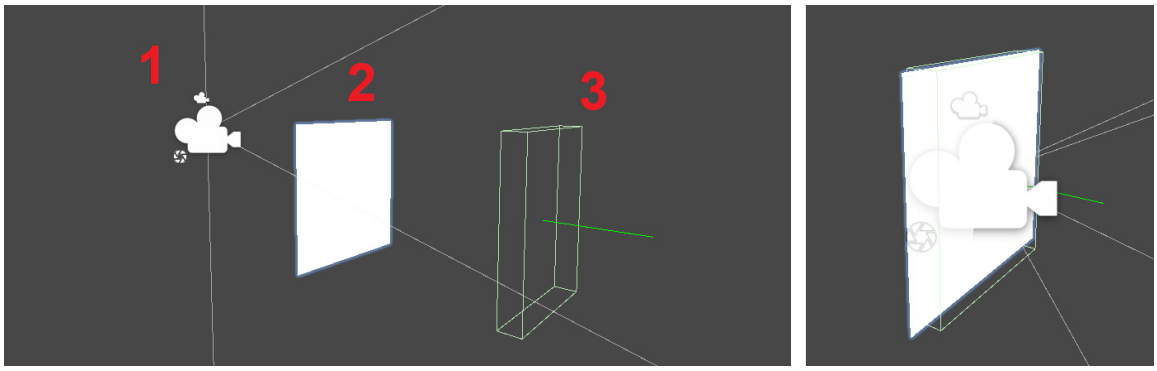
1. vykresľovanie do textúr - portály
2. využitie stencilového bufferu na maskovanie pixelov - effekt identický obrázku 3.6

Nasledujúce podkapitoly detailne popisujú ako jednotlivé efekty boli dosiahnuté.

4.7.1 Implementácia portálového efektu

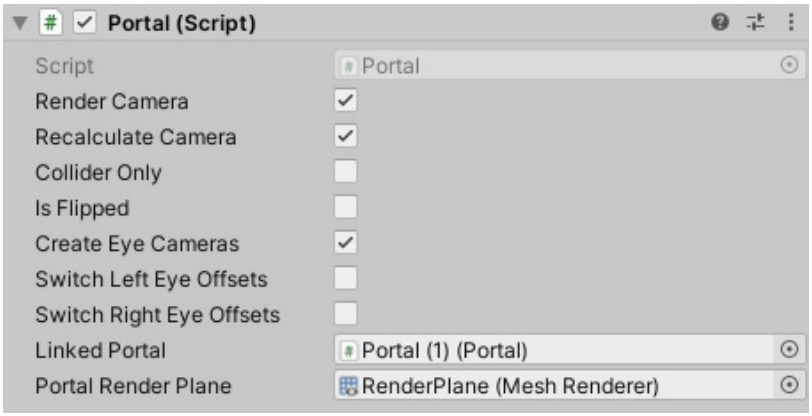
Vykresľovanie do textúr je jednou z najzákladnejších a najužitočnejších metód na tvorbu rôznych efektov v počítačovej grafike. Základnou myšlienkou je vykresliť si scénu z pohľadu sekundárnej kamery do špecifického, užívateľom definovaného bufferu naývaným framebuffer. Tento postup sa môže použiť ci už na tvorbu vodnej hladiny, zrkadiel, alebo aj samotných efektov portálu [5].

Samotný portál v scéne je tvorený herným objektom. ktorý má ako komponentu skript **Portal.cs**, zároveň má tento herný objekt troch potomkov. Na obrázku 4.9 je štruktúra tohoto portálu.



Obr. 4.9: Na ľavo komponenty portálu, na pravo výsledný portál. Jednotlivé komponenty sú nasledovné: 1 - reprezentuje kameru, 2 - vykresľovaciu plochu, 3 - teleportačný collider.

Skript **Portal.cs** je najdôležitejšou komponentou celej konštrukcie. Predstavuje totiž základné nastavenie portálu, vykonáva prepočet pozície kamery portálu a generovanie dodatočných kamier. Po priradení daného skriptu ku hernému objektu sa v Unity Editore zobrazia nastavenia na obrázku 4.10.



Obr. 4.10: Skript **Portal.cs** a nastavenia, ktoré sa môžu meniť podľa potreby.

Jednotlivé nastavenia majú nasledovný význam:

- Render Camera - zapne/ vypne vykreslovanie kamery do textúry
- Recalculate Camera - zapne/ vypne prepočet pozície kamery
- Collider Only - zapne/ vypne vykresľovaciu plochu portálu
- Is Flipped - ak je nastavená hodnota **true**, v metóde **Start()** dôjde k rotácii vykresľovacej plochy a teleportéru o 180 stupňov po osi Y
- Create Eye Cameras - zapne/vypne vytváranie kamier pre pravé a ľavé oko headsetu
- Switch Left Eye offset - invertuje hodnotu offsetu ľavého oka
- Switch Right Eye offset - invertuje hodnotu offsetu pravého oka
- Linked Portal - reprezentuje portál, ktorého pohľad sa bude vykresľovať na Portal Render Plane
- Portal Render Plane - vykresľovacia plocha tohoto portálu

Z nastavení je zrejmé, že je možnosť vytvoriť portály tak, aby v kuse nevytvárali kamery alebo aby sa neprepočítavala ich pozícia. Vytváranie kamier a ich prepočet je totiž pomerne drahá záležitosť, ktorá može značne ovplyvniť FPS[4] aplikácie. V prípade, že je vytváranie kamier a ich prepočet vypnutý ale je aktívny teleportér, portál stále bude fungovať a prenášať hráča z jedného miesta na druhé, či už pri klasickom priechode alebo teleportačným lúčom. Je však nutné effekt neeuklidovského priestoru dotvoriť rozumným plánovaním levelu 4.10, to totiž bude vo väčšine prípadov niekoľko násobne rýchlejšie.

4.7.2 Tvorba dodatočných kamier

Bohužiaľ jedna kamera na vytvorenie portálového efektu je vo VR nepostačujúca. Je nutné pred prepočtom pozície hlavnej kamery vytvoriť ešte dve dodatočné kamery pre každý portál. Tieto dodatočné kamery reprezentujú jednotlivé oči hráča a v hierarchii sa pridajú ako potomkovia portálovej kamery, na ktorú sa hráč pozerá. Za samotné tvorby kamier je zodpovedná implementovaná metóda **CreateEyeCamera(**Camera.StereoscopicEye eye**)**, kde

[4]počet snímkov vykreslených za sekundu, skratka z anglického výrazu „Frames Per Seconds“,

ako vstupom funkcie je informácie o oku, pre ktoré sa má daná kamera vytvoriť. Fukcia najprv vytvorý herný objekt s názvom oka „LeftPortalEye“, „RightPortalEye“, tomuto objektu priradí komponentu kamera a okopíruje nastavenia hlavnej (hráčovej) kamery. Je potrebné správne nastaviť projekčnú maticu takto vytvorených kamier. Na to poslúži metóda **GetStereoProjectionMatrix**(eye). Poslednú vec ktorú je potrebné pre takto vytvorené kamery nastaviť je orezávacia maska (anglicky „culling mask“). Pre takto vytvorené kamery totiž nechceme vykreslovať objekty, ktoré majú nastavenú vrstvu „PortalPlane“.

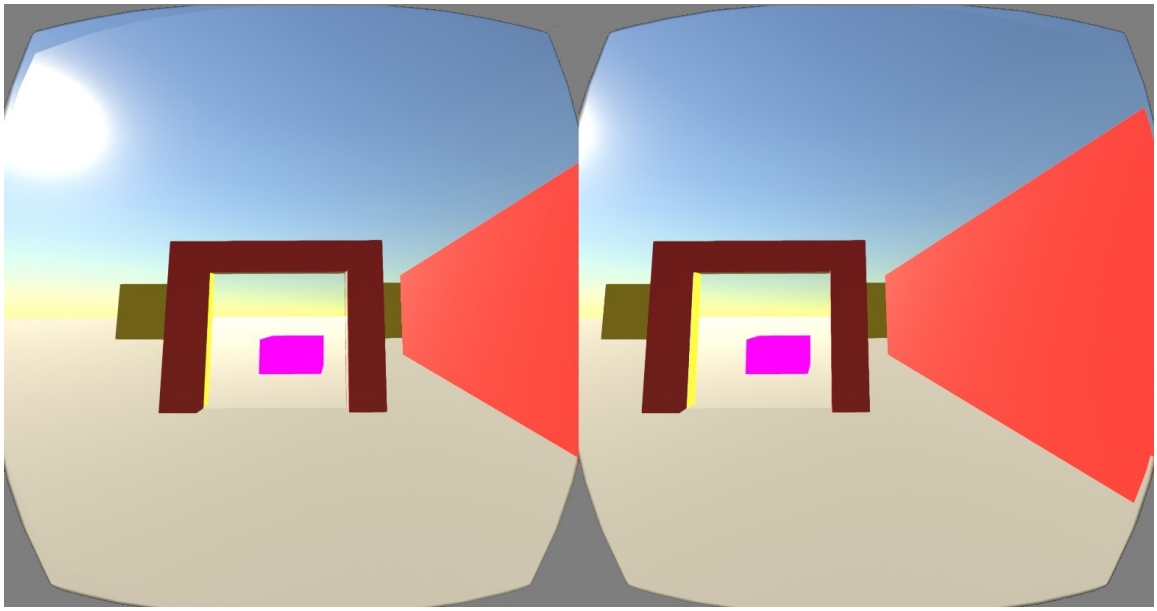
4.7.3 Prepočet novej pozície

Ešte pred samotnou tvorbou kamier z predošlej podkapitoly a nastaveným ich novej pozície je však nutné si zistiť, či sa vôbec dané kamery majú generovať, pozície prepočítavať a vykresľovanie má uskutočniť. Na riešenie problému boli implementované dve funkcie, ktoré prevádzajú iba jednoduchú kontrolu hráča pred samotným prepočtom a kreslením. Obe funkcie, **IsInFrontOfPortal**(...) a **IsInCameraFrustum**(...), očakávajú ako argumenty plochu, na ktorý sa textúra portálu bude nanášať a hlavnú kameru. Funkcia **IsInFrontOfPortal**(...) potom prevádza kontrolu na základe pozície kamery a **forward** vektorom plochy. V prípade, že sa nachádza kamera pred plochou, postupuje sa funkciou **IsInCameraFrustum**(...), kde sa kontroluje či sa daná plocha nachádza v kamerovom fruste. V prípade že sa jedna z funkcií vyhodnotí neúspešne, logicky sa k prepočte nedopracujeme ale zároveň sa aj vymažú kamery vytvorené v predošlom kroku. V prípade úspechu funkcií sa pokračuje prepočtom, kde sa získa relatívna pozícia hlavnej kamery ku portálu a táto pozícia je potom transformovaná do svetových súradníc podľa **transformu** prepojeného portálu, obdobným spôsobom ako to bolo popísane pri detekcií hitu teleportačného collideru v kapitole 4.6.1. Po správnom umiestnení portálovej „dummy“[5] kamery je nutné správne upraviť ešte pozície kamier pre oči. Tu postačí pre respektívne kamery odčítať pozíciu, ktoré si sledujú potomkovia hlavnej kamery, popísaný v úvode tejto kapitoly. Následne sa vykresľuje do textúr, ktoré sú predané materiálu vytvorenému podľa špeciálneho shader programu **PortalVR.shader**.

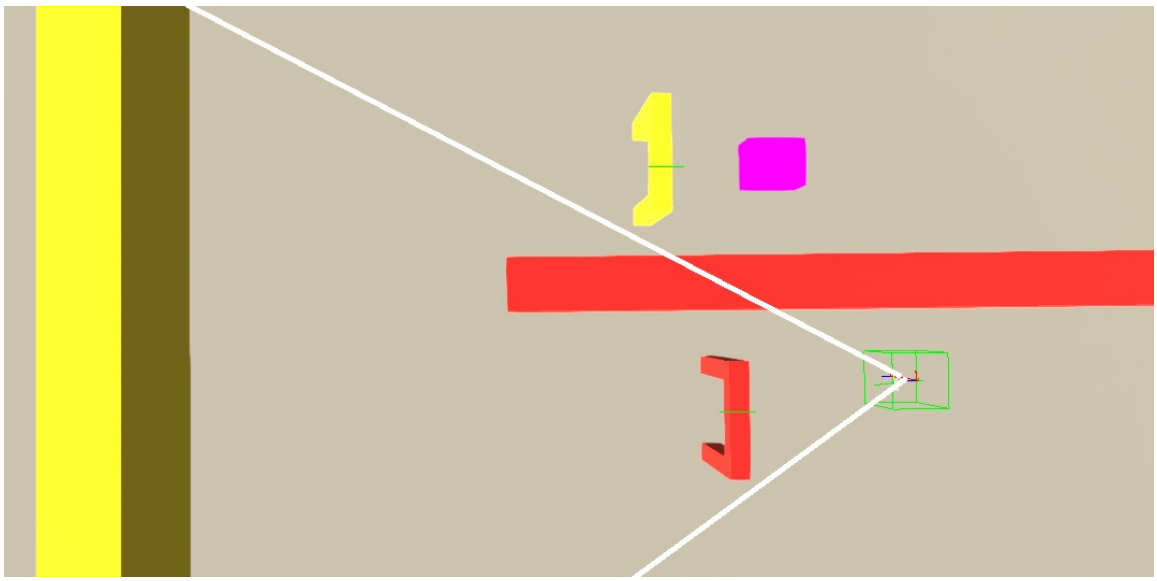
4.7.4 Nanášanie textúr na plochu portálu

Shader program **PortalVR.shader** je zodpovedný na nanesenie textúr vytvorených v predošlom kroku. Vo vertex shadery daného programu sa vykonáva násobenie MVP (model-view-projection) maticou a následne sa pre každý vertex vypočítajú aj jeho súradnice v rámci obrazovkového priestoru (anglicy „screen-space“) pomocou fukncie **ComputeScreenPos**(...). Táto funkcia prevádza jednoduchú konverziu z clip-space súradníc, teda z rozsahu [-1, 1] do screen-space súradníc o rozsahu [0, 1]. Vo fragment shadery sú potom výsledné UV súradnice textúry vypočítané pomocou perspektívneho delenia screen-space súradníc. Textúra je vytvorená pomocou funkcie **tex2D**(texture, uvCoordinates), kde **uvCoordinates** predstavujú spomínané súradnice a **texture**, je textúra, ktorá sa bude používať. Tú je nutné vybrať podľa oka, do ktorého sa práve vykresľuje podľa premennej **unity_StereoEyeIndex**. Ak sa premenná rovná hodnote nula vykresľuje sa pre ľavé oko, ak sa rovná hodnote jedna vykresľuje sa pre pravé. Výsledný efekt portálu pre obe oči je na obrázku 4.11 ako aj pohľad na rovnakú scénu z hora na obrázku 4.12.

[5] kamera z obrázka 4.9, slúži len ako kotva pre kamery očí, nikdy nevykresľuje



Obr. 4.11: Portálový efekt vykreslený pre každé oko zvlášť. V oboch prípadoch je možné si všimnúť dve steny, červenú a žltú. Pri pohľade cez portál ich však nevidieť z dovôdu rotácie napojeného portálu, ktorého pohľad sa vykresľuje na červený portál, kam sa kamera pozerá.

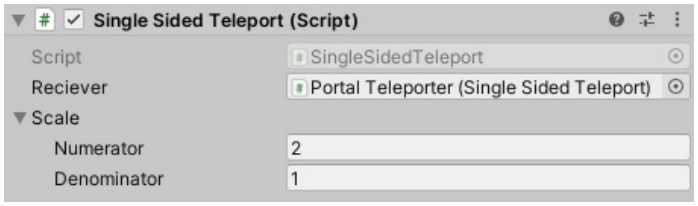


Obr. 4.12: Pohľad z hora na scénu. Zelená kocka reprezentuje hráča a z neho bielymi čiarami je vyznačené kamerové frustum.

4.7.5 Teleportácia

Poslednou komponentou portálu je teleportačný collider. Jedná sa len o jednoduchý herný objekt s nastavenou vrstvou na „PortalCollider“ a priradenou komponentov **BoxCollider**. Pomocou metódy **OnTriggerEnter**(...) je detekovaný moment kedy do **BoxColliderú** vstúpila postava hráča. V prípade, že sa postava hráča nachádza vo vnútri collideru, sa

ustavične prepočítava na ktorej strane plochy, kde sa vykresľuje efekt portálu, sa nachádza. Porovnáva sa hodnota vypočítaná v aktuálnom snímku s hodnotou vypočítanou v predošlom snímku. V momente ak sa hodnoty nerovnajú, došlo k prechodu cez portál a postavu hráča je potrebné premiestniť. Za túto funkcionalitu je zodpovedný skript **SingleSidedTeleport.cs**. Po priradení skriptu, je nutné nastaviť aj koncový **BoxCollider**, ktorý ma taktiež priradený tento skript. Oddelenie samotnej teleportácie od hlavného **Portal.cs** skriptu bol z dôvodu, že niektoré miestnosti sú navrhnuté tak, aby sa samotný efekt portálu nevyužíval ale došlo len ku premiestňovaniu hráča. Zároveň koncový **BoxCollider** vôbec nemusí byť ani súčasťou prepojeného portálu v nadradenom hernom objekte. Toto umožňuje vykresľovanie portálového efektu, ktorý je úplne odlišný ako finálna destináciu teleportácie. Okrem samotného premiestňovania hráča je možné nastaviť, že daná komponenta bude meniť mierku hráča. To je dosiahnuté zadaním premenných **Numerator** a **Denominator**. Na obrázku 4.13 je komponenta **Single Sided Teleport** s priradným koncovým collliderom ako aj s nastavemím zmeny mierky. Zmena mierky sa robí podľa zlomku týchto hodnôt. V prípade príkladu z obrázka 4.13, dôjde teda k dvojnásobnému zväčšeniu mierky hráča.



Obr. 4.13: Komponenta **Single Sided Teleport**.

4.7.6 Stencilový buffer a jeho využitie

Na dosiahnutie efektu ako je na obrázku 3.6 bol použitý stencilový buffer. Za pomoci tohoto bufferu sme schopný vytvoriť špeciálne masky, ktoré budú vykresľovať iba objekty zo špecálnym materiálom a ostatné budú zahadzovať. Na vytvorenie samotnej masky postačí jednoduchý shader program. Vertex shader a fragment shader programu nerobia nič, respektíve vykonávajú iba klasické operácie, prepočet maticou MVP (model-view-projection) a zapísanie farby pixelu na výstup. Na aktivovanie stencilového bufferu pre daný program je potrebné vložiť pred blok **Pass**, ktorý obsahuje vertex a fragment shader, nastavenia stencilového bufferu[21]. Podľa týchto nastavení sa potom bude vykonávať porovnávanie a zapisovanie hodnôt do stenclového bufferu. Nastavenia stencilového bufferu pri shader programe stencilovej masky je vo výpise 4.2.

```
1  Stencil
2  {
3    Ref [_stencilMask]
4    Comp always
5    Pass replace
6  }
```

Výpis 4.2: Nastavenie stencilového bufferu pre masku.

Vo výpise 4.2, parameter **Ref[_stencilMask]**, predstavuje hodnotu podľa ktorej sa bude maskovanie vykonávať. Parameter **Comp**, predstavuje operáciu ktorú bude grafická karta pre daný stencilový test vykonávať pre každý pixel. Premenná **Pass**, reprezentuje

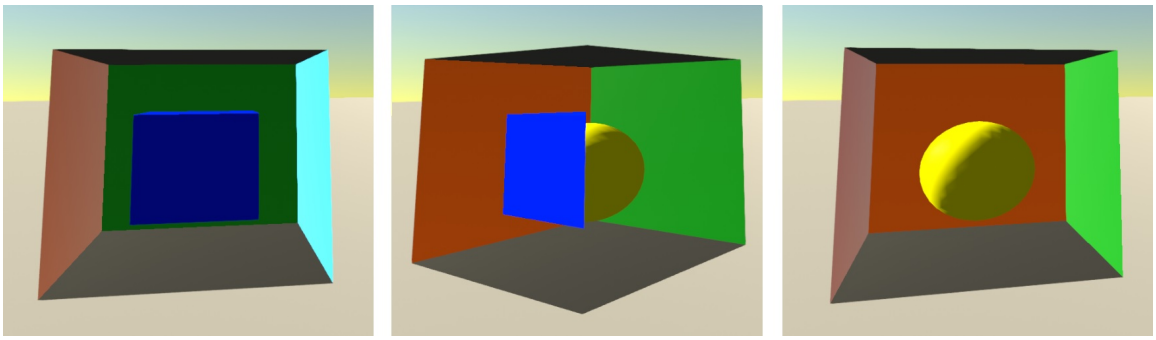
operáciu, ktorá sa má vykonať na stencilovom buffery ak sa úspešne vyhodnotí stencilový a hĺbkový test. Pre kód stencilovej masky dané nastavenia znamená, že pre každý pixel, ktorý prejde týmto shaderom sa vyhodnotí stencilový test úspešne (**Comp always**) zapíše sa hodnota **_stencilMask** do stencilového bufferu (**Pass replace**).

Ďalšou komponentou je materiál, ktorý bude mať taktiež pred blokom **Pass** inicializované nastavenia stencilového bufferu. V tomto prípade obecne stačí zobrať ľubovoľný shader program a rozšíriť ho o potrebné stencilové nastavenia. V tejto práci bola vytvorená kópia shader programu **Legacy Shaders/VertexLit**. Tento shader je oficiálnou súčasťou Unity Enginu. Stencilové nastavenie pre tento shader je:

```
1    Stencil
2    {
3        Ref [_stencilMaterial]
4        Comp equal
5        Pass keep
6    }
```

Výpis 4.3: Nastavenie stencilového bufferu pre materiál.

Grafická karta porovná aktuálnu hodnotu stencilového bufferu voči zadanej hodnote **_stencilMaterial**. Premenná **Comp equal**, hovorí, že daný pixel sa vykreslí v prípade ak hodnota v stencilovom buffery je identická s hodnotou **_stencilMaterial**. Materiál hodnotu v stencilovom buffery prepisovať nepotrebuje, hodnota v stencilovom buffery je teda ponechaná nastavením **Pass keep**. Na obrázku 4.14 je dosiahnutý popísaný efekt.



Obr. 4.14: Jediná kocka v scéne. Kocka je zložená zo šiestich stencilových masiek, na každú stenu kocky jedna maska s rôznou hodnotou na kontrolu stencilového testu. V tejto kocke sa nachádza šesť rozličných objektov, každý objekt z rôznym stencilovým materiálom. V pravej časti má vnútorná kocka identickú hodnotu stencilového materiálu zo stencilovou stenou. V strede je možné vidieť efekt, v momente keď pravá stena kocky má odlišne nastavenú stencilovú hodnotu ako je hodnota materiálu vnútornej kocky. V pravo sa vykresľuje stále ten istý objekt, avšak vnútorná kocka je úplne vymaskovaná a zobrazuje sa guľa.

4.8 Užívateľské rozhranie aplikácie

Užívateľské rozhranie je neoddeliteľnou súčasťou každej aplikácie. Správne navrhnuté užívateľské rozhranie môže prácu s aplikáciou značne zjednodušiť a nesprávne navrhnuté rozhranie naopak môže užívateľa úplne odradiť od samotného produktu. Nie je tomu inak ani

pri aplikáciách vo virtuálnej realite. Princíp tvorby užívateľského rozhrania pre VR aplikáciu je však trošku odlišný od klasických aplikácii vykresľovaných čisto na monitor. Pri vývoji rozhrania musíme dbať na fakt, že sa nám scéna vykresľuje dva krát, raz pre každé oko. Zároveň nemáme prístup ku kurzoru myšky za pomoci ktorého by sme rozhranie mohli ovládať, tak ako u väčšiny aplikácii. V Unity Engine je na tvorbu užívateľské rozhrania dostupných niekoľko systémov:

- UI Toolkit
- Unity UI package (uGUI)
- Immediate Mode Graphical User Interface (IMGUI)

Najnovším systémom je **UI Toolkit**, ktorý okrem samotného rozhrania aplikácie umožňuje rozširovanie aj Unity Editoru. O niečo starším, ale stále relevantným systémom, je objektovo založený **Unity UI package (uGUI)**. Posledným spomenutým systémom je **Immediate Mode Graphical User Interface (IMGUI)**, ktorý umožňuje vytváranie elementov rozhrania za pomoci skriptovania a funkcie **OnGUI**. Slúži primárne na tvorbu vlastných, špecializovaných, pohľadov *Inspector* pre jednotlivé skripty.

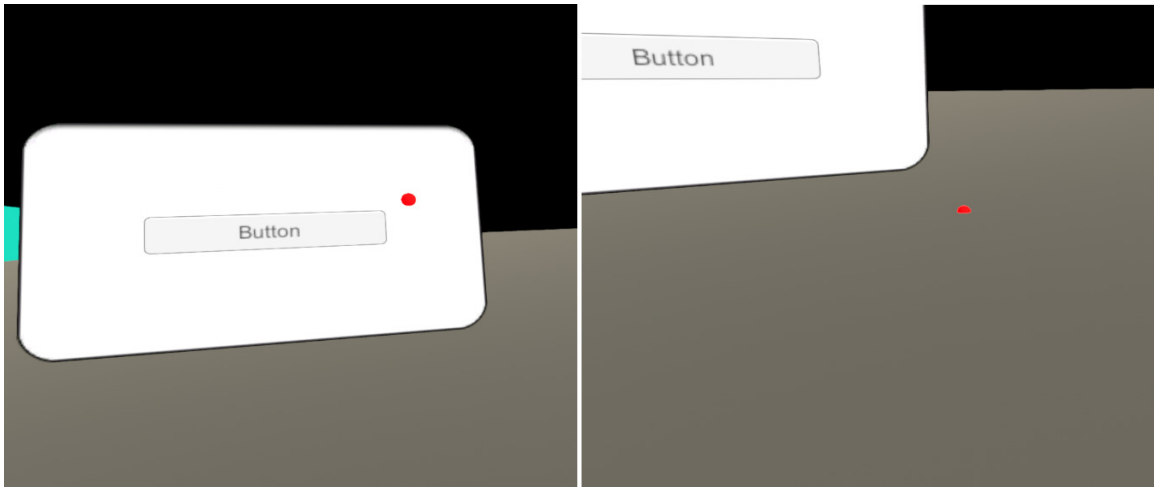
Na tvorbu užívateľského rozhrania bol použitý **Unity UI package (uGUI)**. Princípom je pridať do scény herné objekty, ktoré budú obsahovať špeciálne komponenty užívateľského rozhrania. Základným objektom, ktorý bol do scény pridaný, je objekt s komponentou **Canvas** (ďalej ako „CanvasObjekt“). Vznikajúci problém je, že vykresľovanie tejto komponenty je primárne nastavené do obrazovkového priestoru hlavnej kamery v scéne. To znamená, že užívateľské rozhranie by sa vykresľovalo vždy pred každým okom a vždy pred všetkými ostatnými objektami scény. Inšpiráciou pre riešenie daného problému sa stalo užívateľské rozhranie headsetu Oculus Rift podľa obrázka 4.15

Obr. 4.15: Užívateľské rozhranie headsetu Oculus Rift.

Podobne ako pri užívateľskom rozhraní headsetu Oculus Rift, bolo nastavené vykresľovanie CanvasObjektu priamo do scény podľa jeho svetových súradníc. Toto umožňuje daný

objekt vykresliť obdobne ako všetky ostatné objekty v scéne, teda na danej pozícii, rotáciou a možnosťou meniť mierku. Pozíciu užívateľského rozhrania je potrebné dynamicky prepočítavať podľa pozície a **forward** vektoru kamery hráča. Na výpočet rotácie menu bola použitá technika známa ako bilboarding. Vzidialenosť užívateľského rozhrania od postavy hráča je nastavená implicitne na hodnotu jedna (1), počas hrania má však užívateľ možnosť meniť túto vzdialenosť v menu s nastaveniami.

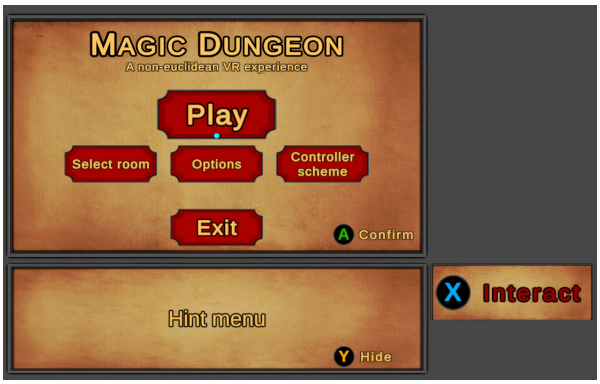
Na rade bolo spracovanie vstupu užívateľa. To prebieha za pomoci **EventSystemu** a metódy **RaycastAll(**PointerEventData eventData, List<RaycastResult> raycastRes**)**, kde **eventData** sú data asociované s kurzorom a **raycastRes** je zoznam všetkých zasiahnutých objektov. Metóda **RaycastAll(...)**, vystreľuje do scény každý základný typ lúču v Unity Engine (**GraphicsRaycast**, **PhysicsRaycast**, **Physics2DRaycast**), z pozície kamery a smeru, ktorým je otočená. Toto je pomerne dôležité, pretože užívateľovi chceme počas manipulácie s UI poskytnúť vždy presné informácie o pozícii kurzoru. Jednotlivé lúče totiž robia detekciu zásahu nad rôznymi elementami scény. Na detekciu zásahu nad UI elemetami sa používa **GraphicsRaycast**, zatiaľ čo **PhysicsRaycast** a **Physics2DRaycast** robia detekciu nad fyzickými objektami v scéne. Na obrázku 4.16, je možné vidieť dva prípady kde dôjde ku zásahu vystrelovaných lúčov a k vykresleniu kurzora. V ľavej časti obrázka sa zasiahol práve CanvasObjekt za pomoci **GraphicsRaycast**, v pravej sa zasiahla podlaha za pomoci **PhysicsRaycast**.



Obr. 4.16: Správne vykreslenie kurzora. V ľavej časti detekciu zásahu za pomoci **GraphicsRaycast**. V pravej časti bola zasiahnutá podlaha vďaka **PhysicsRaycast**.

Kompletné užívateľské rozhranie v aplikácii tvoria dokopy tri CanvasObjekty: hlavné menu, menu s pomocou, interakčné menu. Na obrázku 4.17 je vidieť všetky tri typy CanvasObjektov. Hlavné menu je zo všetkých najkomplexnejšie, obsahuje základné ovládacie prvky aplikácie ako spustiť hru, vypnúť hru, výber miestností a nastavenia aplikácie (hlasitosť jednotlivých efektov, uhol pevnej rotácie, vzdialenosť menu od kamery). Menu s pomocou sa zobrazuje len v istých prípadoch, keď je nutné hráčovi pomôcť. Tento jav je spúšťaný vždy pri vstupe do novej miestnosti s iným textom. Interakčné menu sa zobrazuje v prípade, že hráč našiel objekt označený vrstvou **InteractonCollider** a snaží sa s ním interagovať za pomoci interakčného lúča. Na pozadie menu bol použitý prevzatý obrázok

obyčajného pergamenu[6], zatiaľ čo tlačidlá, okraje a obrázky talčidiel ovládača Xbox boli vytvorené v kresliacom nástroji **GIMP** [7].



Obr. 4.17: Jednotlivé CanvasOjekty, ktoré tvoria kompletné užívateľské rozhranie

4.9 Grafika a design aplikácie

Aby samotná aplikácia pôsobila profesionálnejšie a naozaj dávala dojem počítačovej hry, bolo potrebné ju vhodne obohatiť, či už o rôzne efekty, časticové systémy alebo modely. Samotné modelovanie, animovanie a design je veľmi často odvetvie samo o sebe. Programátori sa obvykle starajú o jednotlivé systémy hry, zatiaľ čo animátory a modelári sa starajú o výzor systémov. Aj keď toto rozdelenie stále platí aj dnes, práve s rozrastujúcou sa popularitou herných enginov, vzniká čoraz viac a viac tzv. indie [8] hier, dosť často vyvíjaných kompletne jednou osobou. Grafika týchto indie hier je veľmi často v štýle zvanom „Low-Poly“[9]. Keďže samotná aplikácia pozostáva z uzavretých úrovní, podkapitola 4.10.1, zvolenou grafickou témou sa stal kúzelný stredoveký žalár. V nasledovných podkapitolách je popis ako daný výzor aplikácie bol dosiahnutý v tejto práci.

4.9.1 Modely a textúry

Po zvolení témy bolo potrebné vytvoriť jednotlivé modely. Na modelovanie bol použitý program **Blender 2.92.0**[10]. Keďže sa jedná o štýl „Low-Poly“, jednotlivé modely nie sú značne komplikované. V prvom rade bolo nutné vymodelovať samotné miestnosti na jednotlivých úrovniach. Na tvorbu miestností bol použitý dosť netradičný spôsob. Najprv boli vymodelované tehly a k nim potrebná textúra, obrázok 4.18. Takto vytvorené tehly boli vyexportované do typu **FBX** a následné nahrané do jednotlivých úrovní. Tam boli modely tehál duplikované a umiestnené, miesnosť bola vykachličkovaná desiatkami až stovkami malých tehál. Po navrhnutí miestnosti bol za pomoci Unity nástroja **FBX Exporter** vyexportovaný kompletný mesh miestnosti, ktorý mohol byť následne v programe **Blender** zjednodušený. Tento proces bol značne zdĺhavý, avšak bol využitý práve kvôli efektom portálov, kde bolo jednoduchšie vytvoriť rozloženie jednotlivých kachličiek v miestnosti

[6] obrázok dostupný na https://www.nicepng.com/ourpic/u2e6y3t4u2a9t4w7_scroll-scroll-parchment-paper-png/

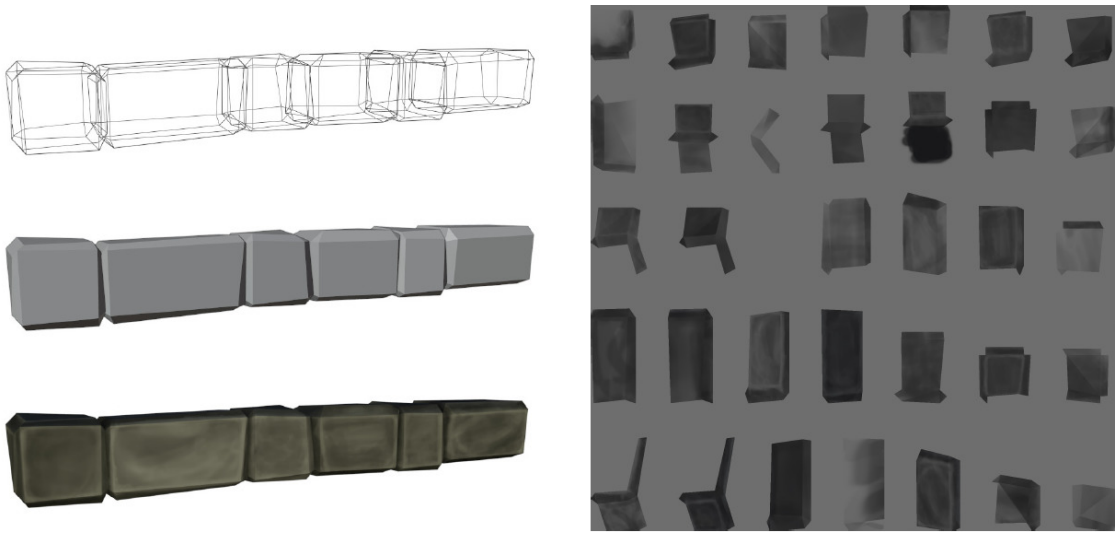
[7] open source softvéer na editáciu obrázkov, dostupný na https://www.gimp.org/

[8] skratka z anglického výrazu „independent game developer“

[9] štýl, kde samotná geometria obsahuje pomerné málo polygonov (skratka z anglického „low on polygons“)

[10] open source softvér na modelovanie a vykresľovanie, dostupny na https://www.blender.org/

„in-real-time“ ako keby sa namodeloval kompletný mesh samostatne, nahral sa do úrovne a v prípade ak by nesedel by sa musel opakovať celý proces modelovania.



Obr. 4.18: Modely tehál a k nim vytvorená textúra

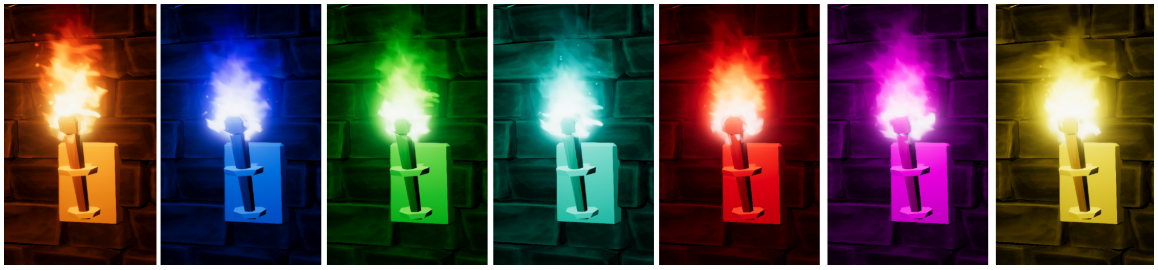
Okrem miestností bol v rámci práce vymodelovaných aj zopár dodatočných modelov reprezentujúcich rôzne veci, ktoré spadajú pod zvolenú tému. Tieto modely sú porozmiestňované po jednotlivých úrovniach. Ukážky modelov, sú v prílohe B.

4.9.2 Osvetlenie a časticové systémy

Jednotlivé úrovne bolo potrebné vhodne osvetliť. Unity Engine umožňuje do scény vložiť smerové svetlo, reflektory, bodové svetlá alebo aj plošné svetlo. Do jednotlivých úrovní boli nakoniec vložené jednoduché bodové svetlá. Tieto bodové svetlá reprezentujú pochodne, ktoré sú pripevnené na stenách. Vložením svetlám bol potom priradený skript **LightFlicker.cs**. Je to jednoduchý skript, ktorý vygeneruje náhodne dvadsať hodnôt a postupne medzi nimi iteruje a mení podľa nich intenzitu svetla.

Obr. 4.19: Textúra plameňov pre časticové systémy

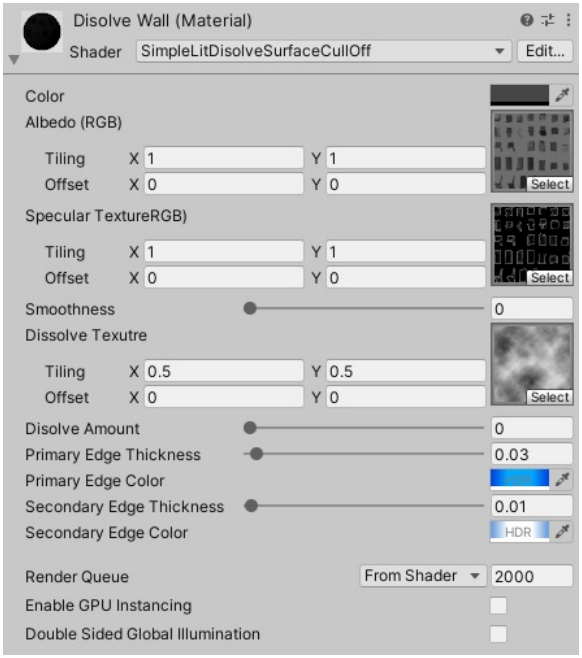
Na plameň pochodní boli použité časticové systémy. Hotový plameň sa skladá celkovo zo štyroch časticových systémov: 1 - vonkajší, tmavší plameň, 2 - vnútorný plameň, 3 - iskry, 4 - časticový systém reprezentujúci žiarivosť. Pre časticové systémy bola vytvorená jednoduchá textúra štylizovaného plameňa 4.19. Zapnutím vlastnosti **Texture Sheet Animation** a nastavením kachličkovania textúry na 2x2 časticový systém danú textúru „rozdelý“ na 4 časti a náhodne vyberie z ktorej časti sa bude daná častica textúrovať. Ďalšími zapnutými vlastnosťami boli zmena veľkosti a farby počas života častice (**Size over Lifetime**, **Color over Lifetime**). Nakoniec bolo vytvorených niekoľko prefabrikácií, zložených zo samotného modelu pochodne, osvetlením a časticovými systémami v rôznych farbách, viz obrázok 4.20.



Obr. 4.20: Vytvorené prefabrikácie pochodní. Každá prefabrikácia obsahuje rovnaké audio, bodové svetlo a časticové systémy. Jedinou zmenou bola vlastnosť **Color over Time**.

4.9.3 Efekt rozpustenia a efekt bariéry

Aby sa v scéne nenachádzali len statické objekty, boli do nej pridané aj jednoduché efekty. Zvolenými efektami sa stal efekt rozpustenia (anglicky „disolve“) a efekt bariéry. Samotné efekty nie sú komplikované a na internete je dostupných nespočetne veľa návodov ako dané efekty vytvoriť. Nevýhodou je, že všetky návody sú vytvorené v **ShaderGraphu**. Jedná sa o vizuálny jazyk na vytváranie shaderov. Bohužiaľ, ShaderGraph je podporovaný iba vo vykresľovacích reťazcoch URP a HDRP. Požadované efekty bolo teda nutné vytvoriť klasickým spôsobom, napísaním si shader programov.



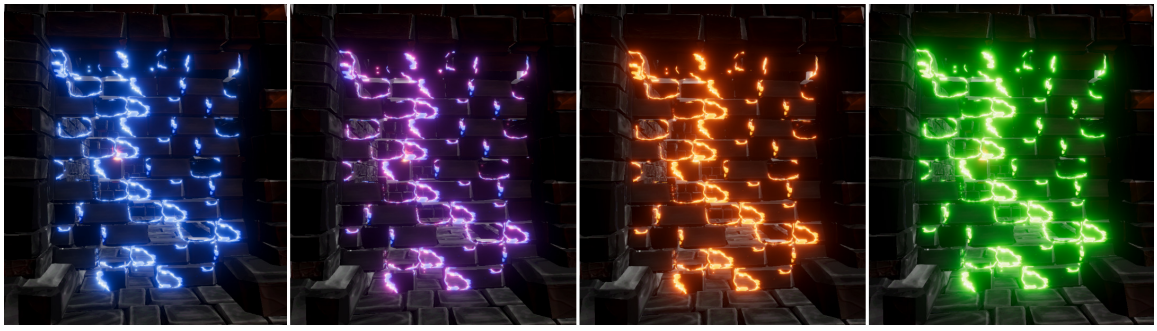
Obr. 4.21: Materiál využívajúci shader na rozpostenie, umožnuje nastaviť farbu a textúru materiálu. Na efekt rozpustenie je potrebné priradiť textúru podľa ktorej sa bude rozpúšťat. Hodnota „DisolveAmmount“ predstavuje mierku ako je materiál rozpustený. Pri prehrávaní efektu sa táto hodnota zvyšuje s plynutím času.

Na efekt rozpustenia je potrebná textúra, pomocou ktorej sa bude objekt rozpúšťať. Na toto postačí jednoduchá textúra šumu. Táto textúra sa predá do shaderu. Podľa intenzity pixelu textúry a intenzity samotného efektu sa potom nastavuje alpha hodnota materiálu

buď na hodnotu jedna alebo mínus jedna. Na vytvorenie okrajov boli použité funkcie **step** a **smoothstep**. S ich využitím je výpočet hrany nasledovný:

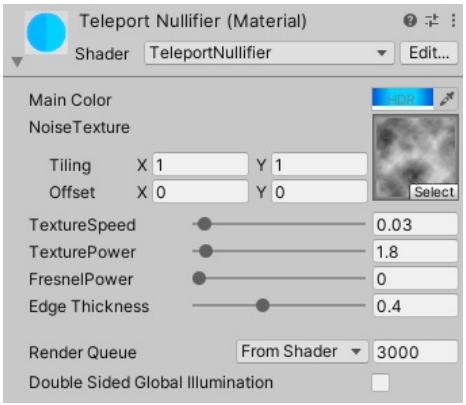
```
1 Edge = EdgeColor*((1.0-smoothstep(0.0,EdgeThickness, DisolveValue))*step(
      DisolveValue,EdgeThickness));
```

Vytvorením viacerých takýchto premenných s rôznou kombináciou hrúbky sme schopný vytvoriť viacero hrán a následne ich skombinovať dokopy jednoduchým sčítaním. Nakoniec je táto novo vypočítaná hodnota priradená do emmisívnej zložky materiálu. Okrem textúry na rozpúštanie umožňuje shader nahrať aj klasické textúry materiálu, umožňuje zmenu farby vonkašej hrany efektu ako aj zmenu vnútornej hrany efektu. Intenzitu samotného efektu môžeme nastaviť staticky alebo nech sa dynamicky mení počas behu aplikácie. Na obrázku 4.21 je možné vidieť materiál vytvorený pomocou tohoto shaderu a na obrázku 4.22 je niekoľko príkladov, ktoré je možno dosiahnuť použitím takto vytvoreného materiálu.



Obr. 4.22: Príklady efektu rozpustenia. Na ľavo vidieť efekt, ktorý má nastavenú vonkajšiu hranu na modrú farbu a vnútornú hranu na bielu farbu. Hneď vedľa je ten samý efekt s jediným rozdielom, farba vnútornej hrany bola nastavená do červena. Zvyšné efekty, slúžia len na demonštráciu, aký má vplyv zmena farby hrán.

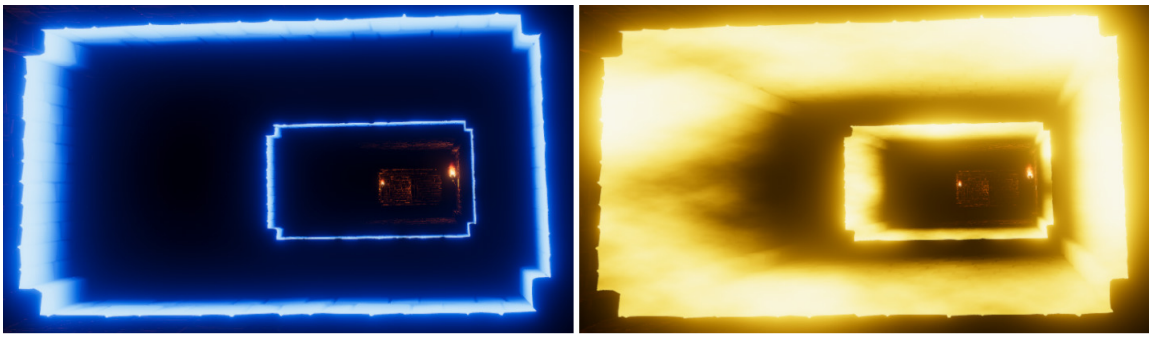
Bariéra, je jednoduchý efekt, ktorý spočíva vo vyznačení prieniku dvoch a viacerých objektov. Na tvorbu efektu bariéry je potrebná hĺbková textúra. Unity Engine umožňuje definovanie takejto textúry priamo v shader programe. V tele shader programu stačí definovať **_CameraDepthTexure**[11]. Premenná **_CameraDepthTexure** bude automaticky naplnená a bude obsahovať potrebnú hĺbkovú textúru. Vo vertex shadery je potrebné si uložiť hĺbku spracovávaného vertexu pomocou **COMPUTE_EYEDEPTH()**. Vo fragment shadery je potom odčítaná táto hodnota od z hĺbokvej textúry. Nová hodnota je prevrátená a vynásobená premennou **EdgeThicknes**, ktorá zodpovedá hrúbke prieniku. Výsledná hodnota je opať nastavená na emisívnu zložku materiálu. Okrem nastavenia farby a hrúbky prieniku, je možné priradiť efektu



Obr. 4.23: Materiál využívajúci shader na bariéru

[11] v starších verziách Unity využiť funkciu UNITY_DECLARE_DEPTH_TEXTURE(depthTexture)

aj textúru a určiť rýchlosť akým sa bude textúra pohybovať, viz obrázok 4.23. Na obrázku 4.24 je efekt bariéry.



Obr. 4.24: Materiál bariéry aplikovaný na jednoduchú kocku, na ľavo základný, na pravo s aplikovanou textúrou šumu a zvýšenou hrúbkou prieniku

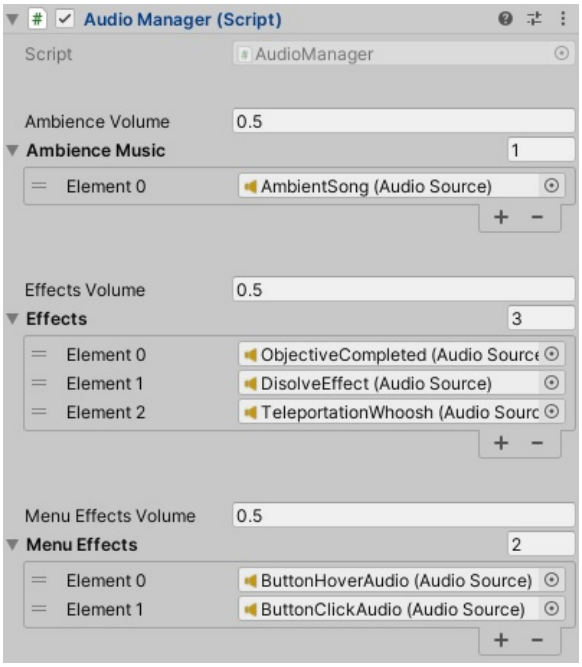
4.9.4 Zvukové efekty

Na dotvorenie atmosféry bolo do aplikácie pridaných aj niekoľko zvukových efektov a ambientnej skladby. Do scény bol pridaný herný objekt a k nemu vytvorený skript **AudioManager.cs**. Tento skript slúži na triedenie jednotlivých hudobných skladieb a na nastavovanie ich hlasitosti. Na obrázku 4.25 je zobrazená komponenta **AudioManager**. Ako je možné vidieť na obrázku, komponenta obsahuje tri kategórie zvukových efektov:

- Ambience
- Effects
- Menu Effects

Tieto kategórie obsahujú herné objekty, a každý z týchto herných objektov má priradenú komponentu **AudioSource**. Komponenta **AudioSource**, obsahuje jednak samotný audio klip, ktorý sa má prehrať ako aj jeho nastavenia. Kategória **Ambience** predstavuje ambientnú skladbu[12], ktorá hrá po celý čas behu aplikácie. Kategória **Effects**, predstavuje špeciálne efekty, ktoré sa prehrajú jednorázovo, napríklad efekt rozpustenia[13], efekt splnenej úlohy [14], teleportácie [15], alebo podobne ako ambientná skladba, hrajú dookola, oheň a praskajúce drevo pochodní [16]. Kategória **Menu Effects**, predstavuje zvukové efekty asociované s užívateľským rozhraním. Jedná sa o vstup, výstup[17] a stlačenia[18] tlačidla na užívateľskom rozhraní.

[12] dostupné na https://freesound.org/people/phlair/sounds/388340/
[13] dostupné na https://freesound.org/people/CosmicEmbers/sounds/387353/
[14] dostupné na https://freesound.org/people/Dneproman/sounds/334765/
[15] dostupné na https://freesound.org/people/qubodup/sounds/60013/
[16] dostupné na https://freesound.org/people/FractalStudios/sounds/363092/
[17] dostupné na https://freesound.org/people/annabloom/sounds/219069/
[18] dostupné na https://freesound.org/people/Nightflame/sounds/422514/



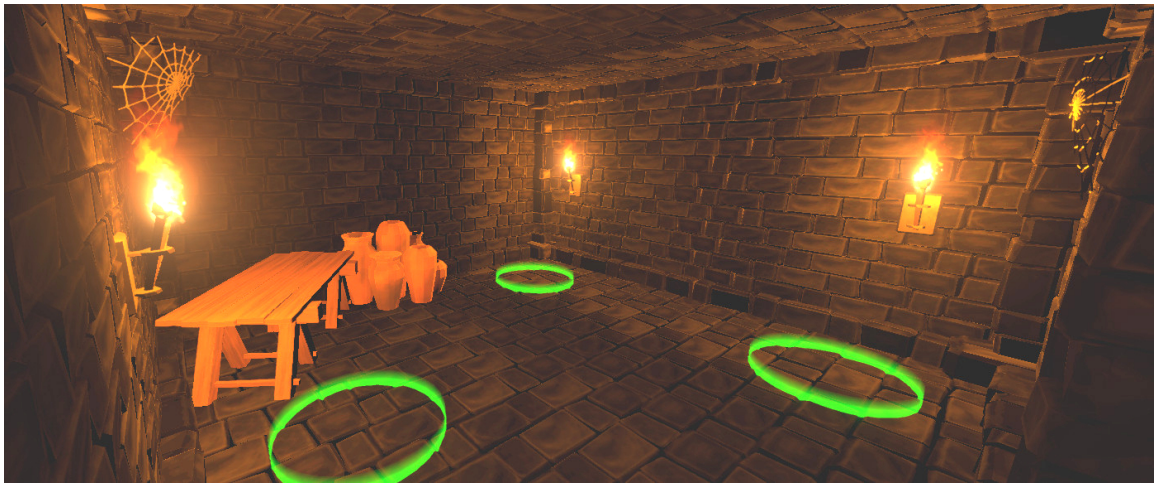
Obr. 4.25: Komponenta **Audio Manager**, s tromi kategóriami skladieb, pre každú kategóriu umožňuje samostatne nastaviť hlasitosť ako aj prehadzovať jednotlivé skladby medzi kategóriami

4.10 Návrh úrovní a ich správa

Aplikácia je rozdelená do jednotlivých úrovní, levelov. Hlavnou pointov týchto úrovní je demonštrácia neeuklidovských effektov dosiahnutých práve samotným rozložením miestností v úrovniach, využitím portálového efektu popísaneho v podkapitole 3.2.1 alebo stencilovej masky z podkapitoly 4.7.6. V aplikácii sa nachádza celkovo desať (10) úrovní, nerátajúc počiatočnú úroveň a koncovú úroveň. Úrovne sú navrhnuté tak, aby mali jeden vchod a jeden východ. Cieľom je dostať sa k východu úrovni, kde správna cesta je vždy iba jedna. Niektoré miestnosti sú štruktúrované tak, že okrem samotného neeuklidovského efektu majú aj menšiu hádanku, úlohu, ktorú je nutné splniť aby sa odhalila pravá cesta. V nasledovných podkapitolách budú bližšie popísané jednotlivé úrovne. Keďže prostredie úrovní je veľmi tmavé, obrázky k nim bolo nutné upraviť zvýšením jasu.

4.10.1 Úroveň 0 - Starting Room

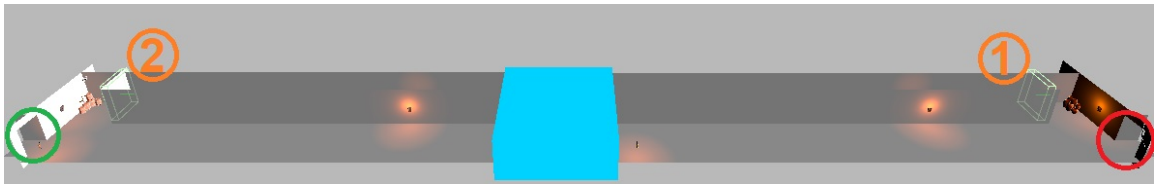
Počiatočná miestnosť logicky predstavuje miestnosť, odkiaľ hráč bude začínať. Hráč je privítaný užívateľským menu, kde má možnosť zahájiť svoju hru, prípadne si pozrieť schému ovládača alebo si nastaviť hlasitosť hudby a zvukových efektov. Jedná sa jednoduchú uzavretú miestnosť. Obrázok 4.26 reprezentuje túto úroveň.



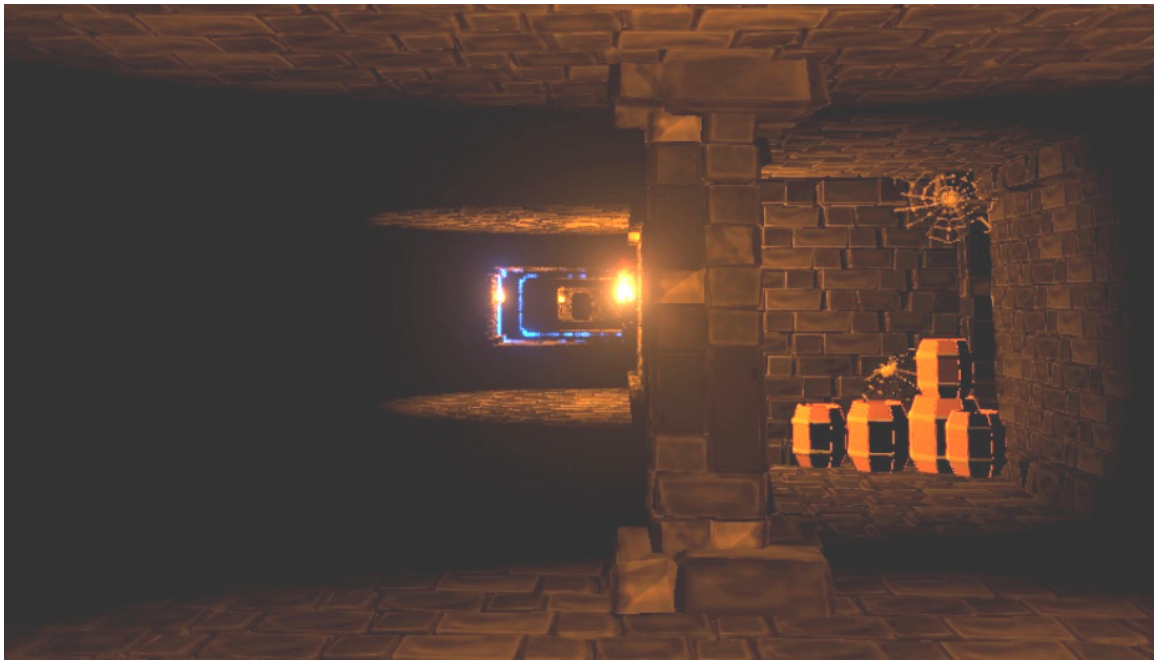
Obr. 4.26: Úroveň nula. Hneď v úvodnej miestnosti hráč narazí na úlohu, ktorú je nutné splniť pre ďalšie pokračovanie. Táto úloha učí hráča pohybu pomocou teleportačného lúča a rotovania sa o pevne daný uhol. Cieľom je vkročiť do vyznačených zelených kruhov. Po vkročení do kruhu sa prehrá zvučka, že bola daná pod úloha splnená. Po navštívení všetkých troch kruhov sa vytvoria dvere do ďalšej miestnosti.

4.10.2 Úroveň 1 - Long Tunnel

Úroveň jedna pozostáva z jedného dlhého tunela a cieľ úrovne je dostať sa na druhú stranu. Na obrázku 4.27 je rozloženie úrovne a na obrázku 4.28 je výsledný neeuklidovský efekt v miestnosti dosiahnutý využitím portálov.



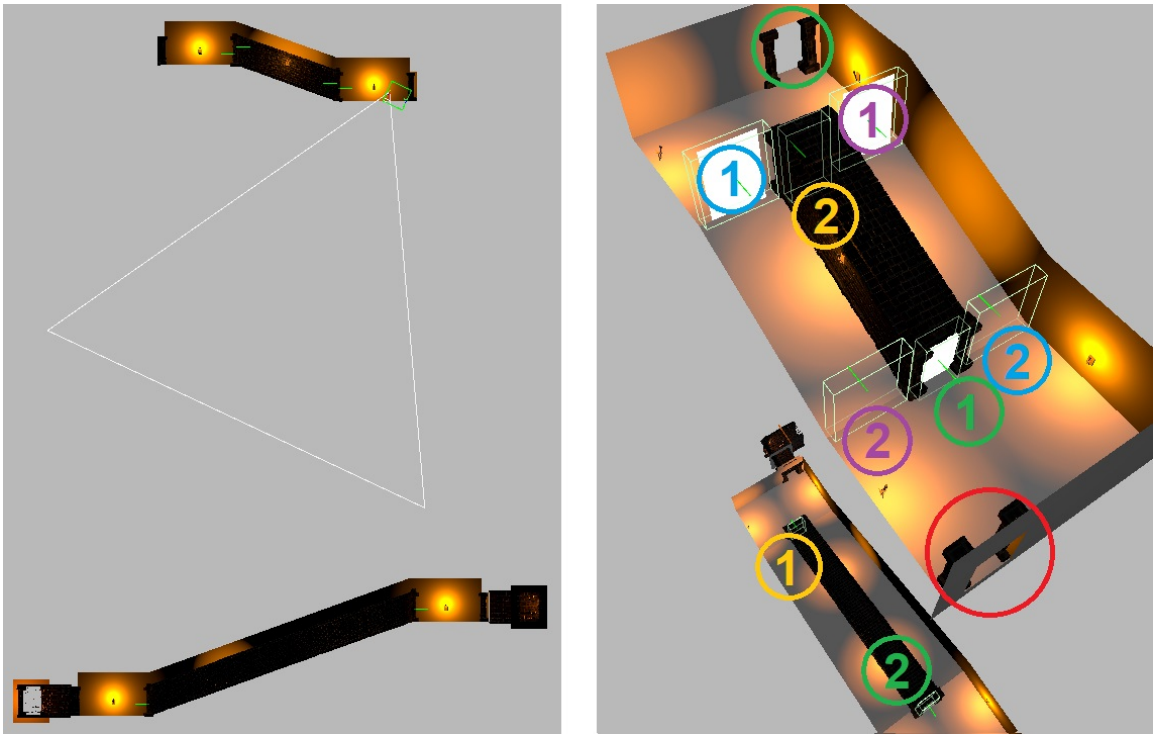
Obr. 4.27: Úroveň jedna. V strede tunela je modrou farbou vyznačná bariéra zabraňujúca prechod. Hráč musí nájsť cestu okolo tejto bariéry. Cesta je vytvorená využitím portálov. Ich využitím taktiež vzniká neeuklidovský efekt, kde samotná cesta cez portály je niekoľko násobne kratšia. Vchod do miesntnosti je vyznačený červeným kruhom, východ zeleným. Prepojené portály sú vyznačené svojím **BoxColliderom** a oranžovými číslami.



Obr. 4.28: Pohľad na hotovú úroveň. V pravej časti vidieť dlhý tunel a v ľavej krátky. V pozadí dlhého tunelu je možné vidieť modrú bariéru. Sudy ktoré vidieť v krátkom tunely, sú v skutočnosti opreté o zadnú stenu pri východe.

4.10.3 Úroveň 2 - Up the hill

Úroveň dva je prvou úrovňou, ktorá je zostrojená z viacerých miestností. Na obrázku 4.29 je zobrazená celá úroveň a v jeho pravej časti portály nachádzajúce sa v scéne. Opäť je cieľom hráča nájsť správnu cestu k východu. Tá vedie cez tunel. Neeuklidovský efekt je dosiahnutý tým, že samotný východ sa nachádza na „kopci“, kde tunel vedúci k nemu smeruje do zeme.



Obr. 4.29: V ľavej časti obrázka vidieť rozloženie miestností v úrovni dva. Toto rozloženie bolo zvolené aby sa zabránilo vykresľovaniu portálov, ktoré hráč nevidí. Na obrázku je vidieť že spodná miestnosť je úplne mimo frusta hlavnej kamery. V pravej časti obrázka je vyznačený vchod do miestnosti červeným kruhom a východ zeleným kruhom bez čísla. V úrovni sa nachádza celkovo 8 portálov. Prepojené portály sú vyznačené spoločnou farbou. Správna cesta je naznačená portálom zelená-1, ktorý prenesia hráča na portál označený ako zelená-2. Odtiaľ sa pokračuje do portálu žltá-1, ktorý prenesie hráča na portál žltá-2.

Obr. 4.30: Výsledný neeuklidovský efekt v úrovni dva. K východu, ktorý je vyznačený v kruhu nevedie priama cesta, je nutné použiť tunel, ktorý však vedie úplne iracionálnym smerom.

4.10.4 Úroveň 3 - Two Rooms

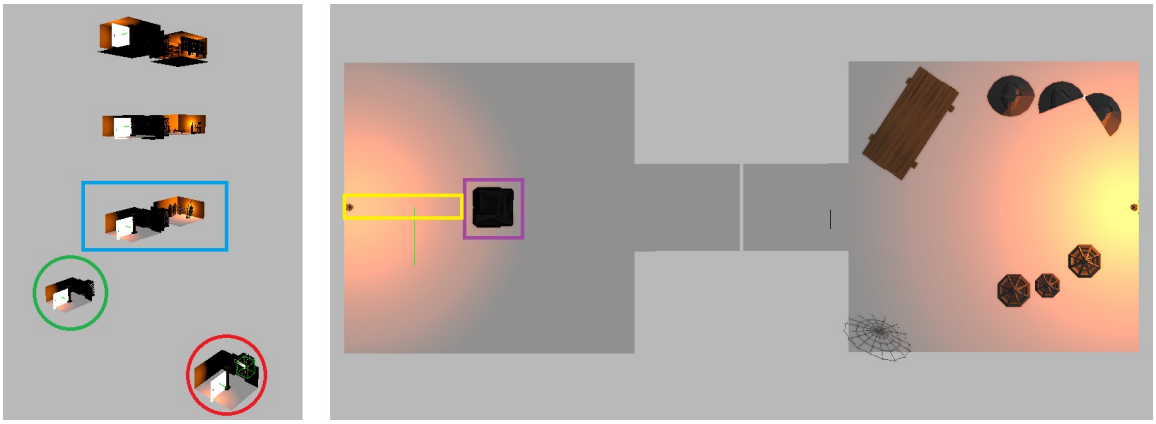
Úroveň tri pozostáva z dvoch identických miestností. Na rozdiel od úrovni dva, ktorá bola navrhnutá aby boli portály neviditeľné pre hráča, v úrovni tri platí presný opak. V strede oboch miestností sa nachádzajú prepojené portály. Miestnosti neobsahujú žiadnu hádanku, hráčovi stačí prejsť priamo cez portál vo vchodovej miestnosti a ocitne sa vo východnej.

Obr. 4.31: Rozloženie miestností v úrovni tri. V strede oboch miestností sú prepojené portály. Portály sú otočené tak, aby hráč pri pohľade cez ne videl vyvesené štíty a pochodeň na stene.

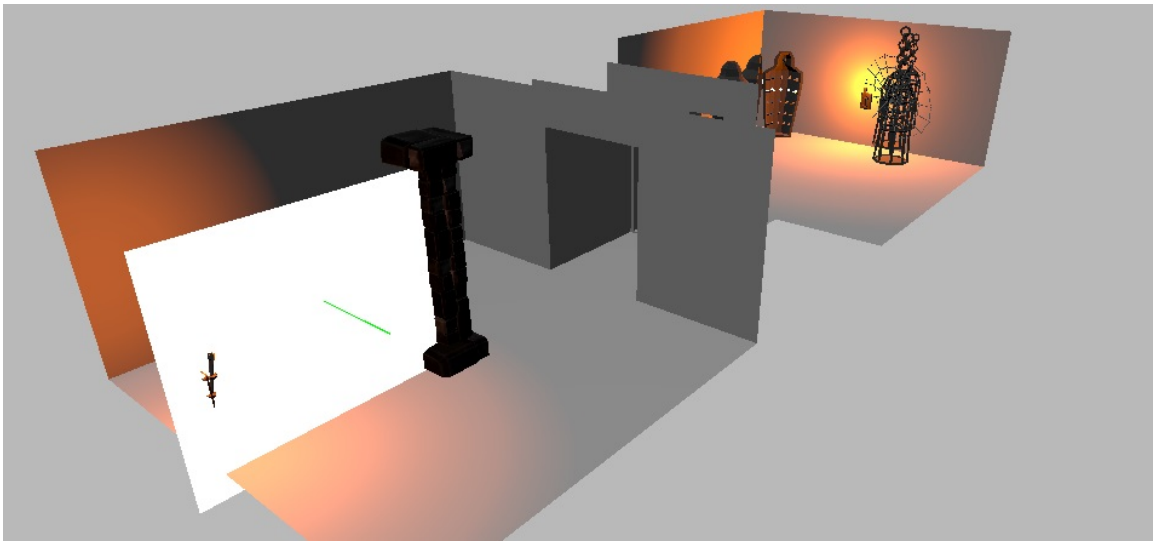
4.10.5 Úroveň 4 - Pillar Rooms

Úroveň štyri je prvou úrovňou, ktorá využíva portály len na samotné premiestňovanie hráča. Dodatočné vytváranie kamier v tejto úrovni je úplne vypnuté. Celá úroveň pozostáva z troch

miestností, vchodu a východu. V každej miestnosti sa nachádzajú dva portály umiestnené na rovnakej pozícii avšak s opačnou rotáciou. Na prechod do ďalšieho podlažia je potrebné prejsť okolo stĺpu nachádzajúceho sa v strede miestnosti. viz obrázky 4.32 a 4.33



Obr. 4.32: V pravej časti obrázka možno vidieť rozloženie jednotlivých miestností do podlaží. Červeným kruhom je vyznačený vchod, zeleným východ. Modrým vyznačené podlažie je zobrazené v pravej časti obrázka. Tam je možné si všimnúť stĺp v strede miestnosti vyznačený fialovým štvorcom. Okolo tohoto stĺpu je nutné prejsť aby sa hráč dostal do portálu, ktorý sa nachádza na mieste vyznačenom žltým kvádrom.

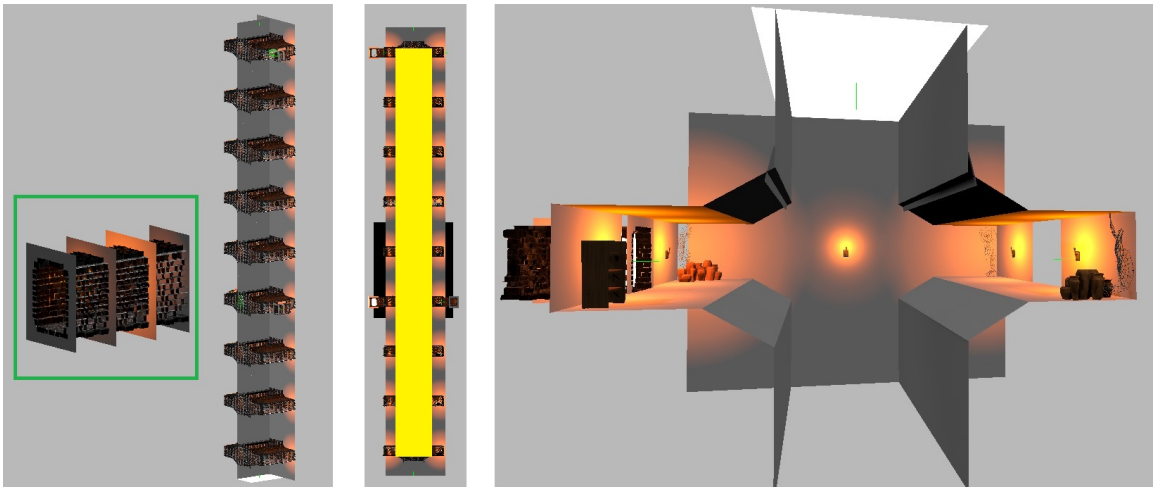


Obr. 4.33: Pohľad na jedno podlažie. Stĺp v strede miestnosti. Biela plocha reprezentuje portál. V zadnej časti miestnosti je možné si všimnúť statické modely vytvorené v programe **Blender**.

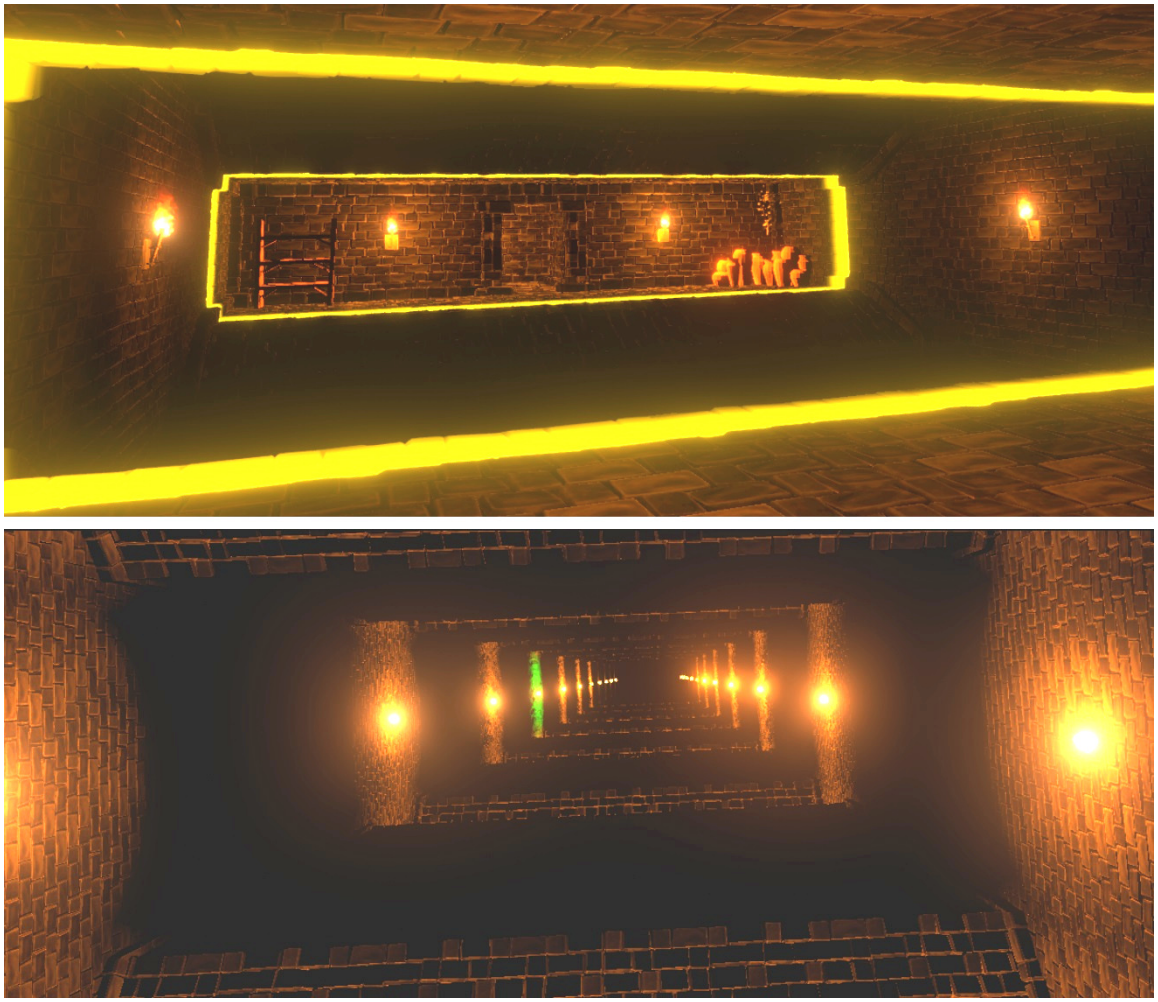
4.10.6 Úroveň 5 - The Descent

Táto úroveň pozostáva z jedného dlhého vertikálneho tunela. Tunel je tvorený duplikovaním prefabrikácie vchodovej miestnosti, obrázok 4.34. Na spodku tunela a vrchole sa nachádzajú prepojené portály. Cieľom je podľa indície nájsť správne podlažie, ktoré obsahuje východ. Všetky ostatné podlažia obsahujú portály, ktoré prenesú hráča na začiatok úrovne. Jedná

sa o najdlhšiu úroveň v hre, z tohoto dôvodu bolo nutné riešiť jej optimalizáciu pridaním komponenty **LOD Group**, ktorá umožňuje na základe vzdialenosti kamery vykresľovať rôzne úrovne detailov modelu miestnosti. Podobne do každej prefabrikácie miestnosti bol pridaný aj **BoxCollder**, ktorý pri vstupe doň, nahrá potrebné portály prepojené z portálom vo vchodovej miestnosti. Keďže sa jedná o vertikálnu miestnosť bolo potrebné pridať **BoxCollider** reprezentujúci gravitačné pole. V ňom má hráč konštantnú gravitáciu.



Obr. 4.34: V ľavej časti vidieť celú úroveň, tak, ako sa skutočne nachádza v scéne. Východ, vyznačený zeleným štvorcom, je v skutočnosti osem krát zväčšený ako zvyšné miestnosti. Dôvodom je mechanika popísaná v podkapitole k úrovni šesť,4.10.7. V strednej časti obrázka je zobrazený pohľad scény kolmý na osu x. V tejto časti vidieť dlhý žltý podtunel, ktorý sa tiahne stredom všetkých miestností. Tento tunel je zodpovedný za upravenie gravitácie hráča. V pravej časti je možné vidieť ako vyzerá prefabrikácia miestnosti.



Obr. 4.35: V hornej časti obrázka je zobrazená hotová prefabrikácie miestnosti a v spodnej časti obrázka je tunel, do ktorého hráč musí skočiť. Zelené svetlo v tunely predstavuje východ.

4.10.7 Úroveň 6 - Scaling Tunnel

Úroveň šesť, opäť pozostáva iba z jednej miestnosti. V miestnosti sa nachádzajú dva portály s odlišnou veľkosťou. Cieľom hráča je nájsť východ s využitím týchto portálov. Pri vstupe do miestnosti je hráč osemnásobne zväčšený. Portály v miestnosti majú špeciálnu vlastnosť, pomocou priechodu cez ne sa veľkosť hráča v miestnosti buď zmenší alebo zväčší.

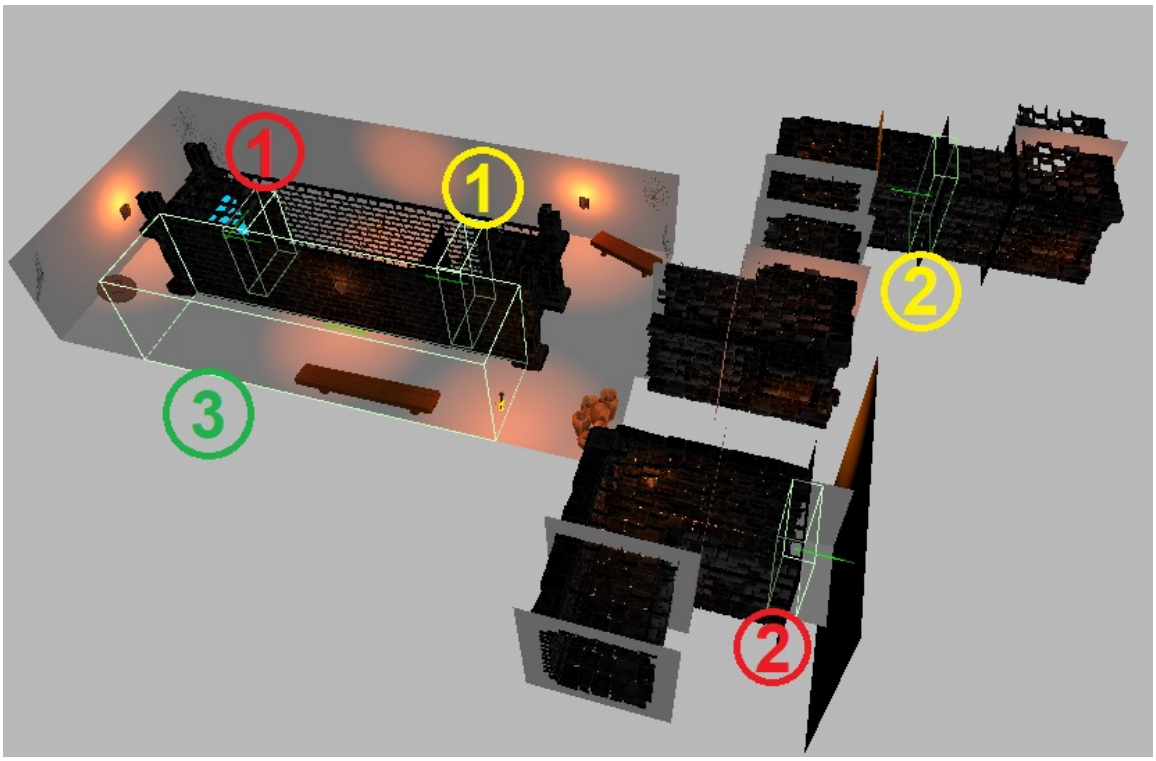
Obr. 4.36: Pohľad do miestnosti v úrovni šesť. Vo vyznačenom zelenom kruhu sa nachádza východ z tejto úrovne.

Obr. 4.37: Priblížený pohľad na východ v úrovni šesť. V jeho okolí možno vidieť tehly, ktoré sú oproti východu osem krát zväčšené.

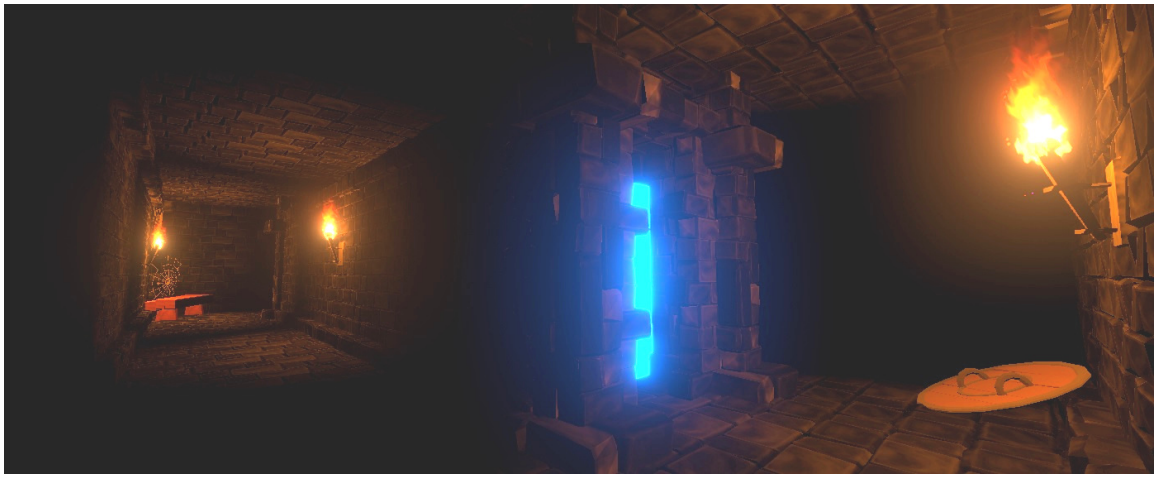
4.10.8 Úroveň 7 - Hidden Tunnel

Siedma úroveň je veľmi podobná úrovni dva v tom, že sa v strede miestnosti nachádza tunel. Pri vchode do tunela spredu je hráč prenesený na začiatok úrovne. Zo zadnej strany tunela sa nachádza stena. Cieľom je nájsť mechanizmus ako stenu odstrániť. V miestnosti

je neviditeľný **BoxCollider**, ktorý sa správa ako jednoduché počítadlo. Pri vstupe do neho sa hodnota buď zvýši alebo zníži, záležiac zo strany vstupu. V prípade že hráč je schopný takto „napočítať“ do hodnoty tri, stena zmizne.



Obr. 4.38: Miestnosť predstavujúca úroveň sedem. Lokácie skrytých portálov sú vyznačené farebne číslami. Prepojené portály majú identickú farbu. Portál červená-1 je skrytý za stenou. Túto stenu je nutné odstrániť. Vyznačený **BoxCollider**, zelená-3, vedľa tunela slúži ako počítadlo koľko krát ním hráč prešiel spredu. Pri každom priechode ním sa odstráni kúsok zo steny.



Obr. 4.39: Pohľad na čiastočne otvorenú zadnú stenu tunela v úrovni sedem. Modrá žiara, vychádzajúca z tunela predstavuje bariéru. Tá je v tunely z dôvodu aby bol hráčovi zabránený vstup do tunela, dokým nebola stena úplne odstránená. Bariéra potom zmizne a hráčovi sa vstup do tunela z tejto strany umožní.

4.10.9 Úroveň 8 - Looped

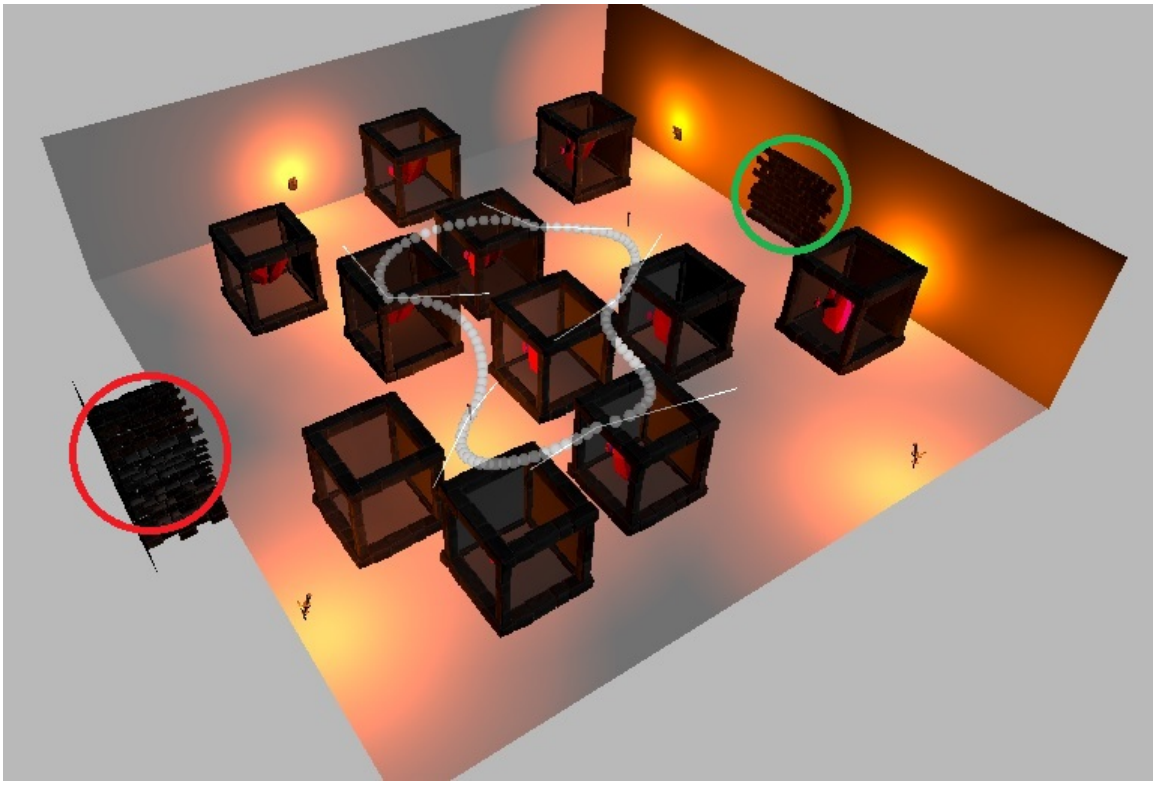
Táto úroveň predstavuje úmysel kombinovania miestností do jednotného priestoru. V úrovni sa nachádza dvanásť miestností rozdelených na tri podlažia. Miestnosti v jednom podlaží sú cyklicky napojených na seba. Neeuklidovský efekt je dosiahnutý tým, že hráč postupne prechádza jednotlivé miestnosti a s využitím portálov je premiestnený do nového podlažia, bez toho aby si to sám uvedomil. V scéne sa teda hráč ocitne na úplne inej pozícií avšak, s pohľadu hráča to vyzerá akoby cyklil neustále dookola. Pri vchode do miestnosti hráč obdrží indíciu, ktorý objekt má nájsť a po interakcii s ním sa mu odhalia dvere do ďalšej úrovne.

Obr. 4.40: Na pravo je možné vidieť jednotlivé podlažia úrovne. V strede rozloženie miestností v podlažiach s vyznačeným portálom. Napravo je pohľad na vyznačený portál.

Obr. 4.41: Dve miestnosti z dvanástich, na ktoré hráč môže natrafiť v úrovni osem.

4.10.10 Úroveň 9 - Puzzle Cubes

V tejto úrovni má hráč za úlohu „poskladať“ si vlastný východ. V miestnosti sa nachádzajú kocky s využitím stencilového bufferu, podkapitola 4.7.6. Vo vybraných kockách sa nachádza časť východu, ktorý hráč musí nájsť a interagovať s ním. Po interakcii sa vybraná kocka premiestni na svoje miest využitím skriptu **MoveObjectScript.cs**. V ňom je možne nastaviť pozíciu kam sa objekt presunie, jeho výslednú rotáciu a čas, ako dlho daná animácia má trvať. Po nájdení všetkých kociek s časťami východu sa hráčovi odhalia dvere. Keďže samotná miestnosť je dosť veľká bolo nutné vymyslieť spôsob akým bude osvetlená. Na osvetlenie bola implementovaná kubická beziérová krivka, ktorú je možné priradiť pochodniam v scéne. Vytvorením viacerých kriviek je možné vytvoriť uzavretú cestu ako na obrázku 4.42.

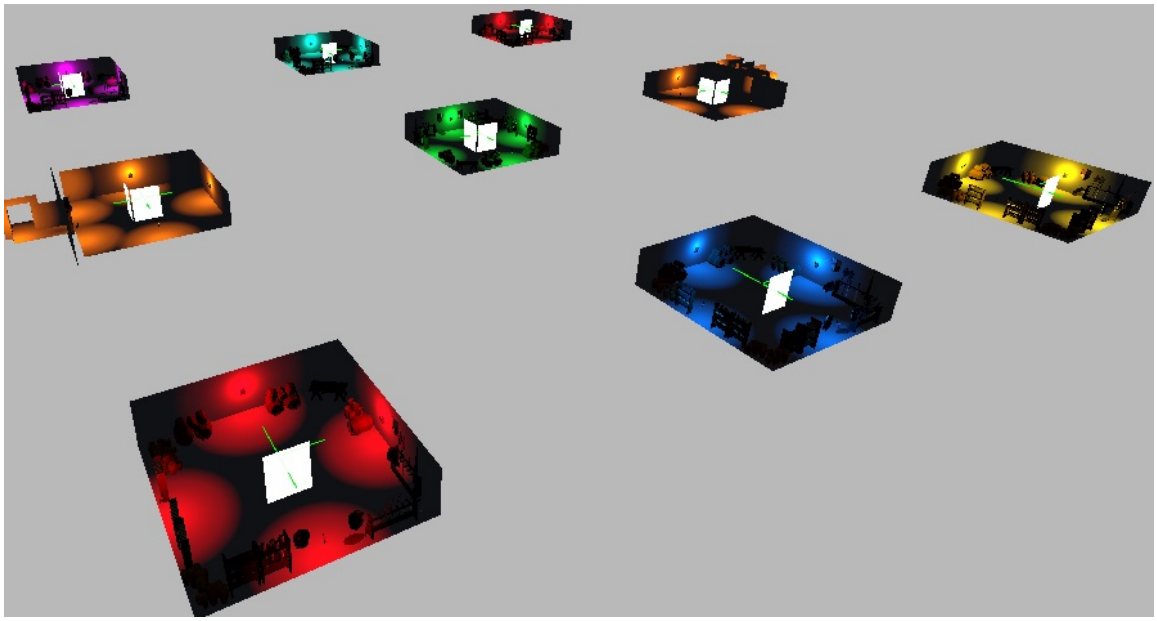


Obr. 4.42: Úroveň deväť. Východ vyznačený zeleným kruhom a vchod červeným. Bielou vyznačená cesta v strede miestnosti, reprezentuje päť beziérových kriviek po ktorých sa pohybujú pochodne v scéne.

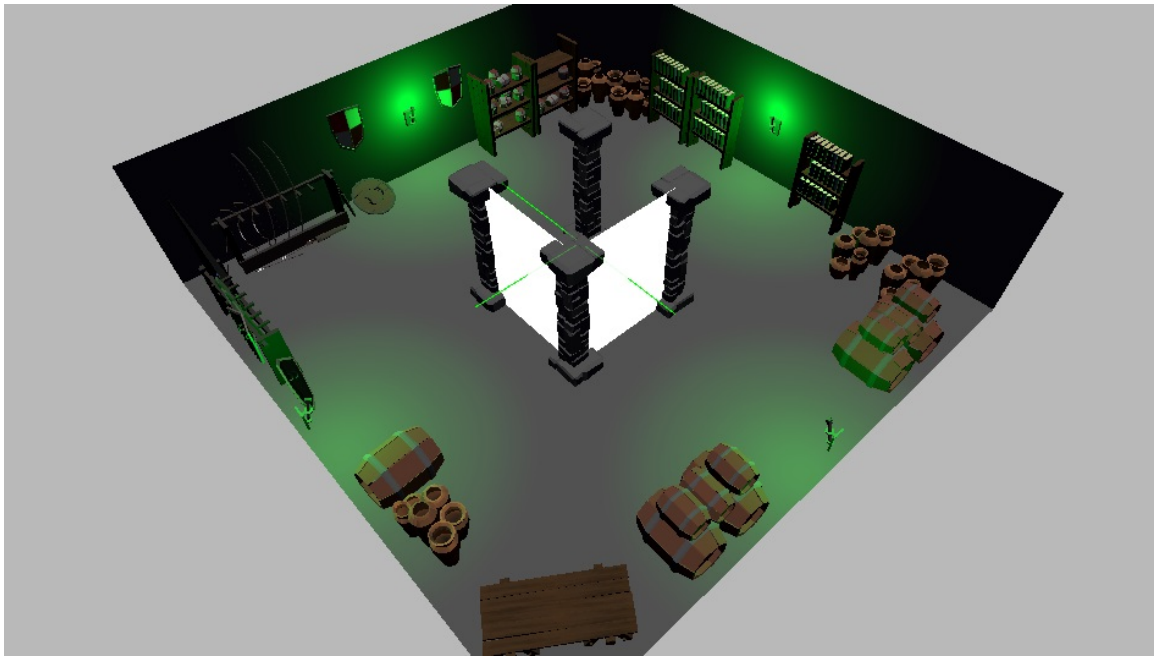
Obr. 4.43: Hotová úroveň deväť. Na obrázku možno vidieť niekoľko kociek vytvorených pomocou stencilového bufferu. V kockách sa nachádzajú kryšály, každý s iným emisívnym materiálom. Steny kociek sú priehľadné.

4.10.11 Úroveň 10 - Rune Labyrinth

Posledná úroveň obsahujúca hádanku. Pri vchode do miestnosti sa hráčovi neposkytne žiadna pomoc. Jediné čo daný hráč môže robiť je prehľadávať jednotlivé miestnosti v úrovni, je ich celkovo deväť. dokým nenarazí na miestnosť s východom. Až pri nájdení poslednej miestnosti vyskočí hráčovi menu s pomocou. To mu poradí aby sa zameral na štyri runy, ktoré sú vykreslené nad východom. Účelom hráča je potom aktivovať tieto runy.

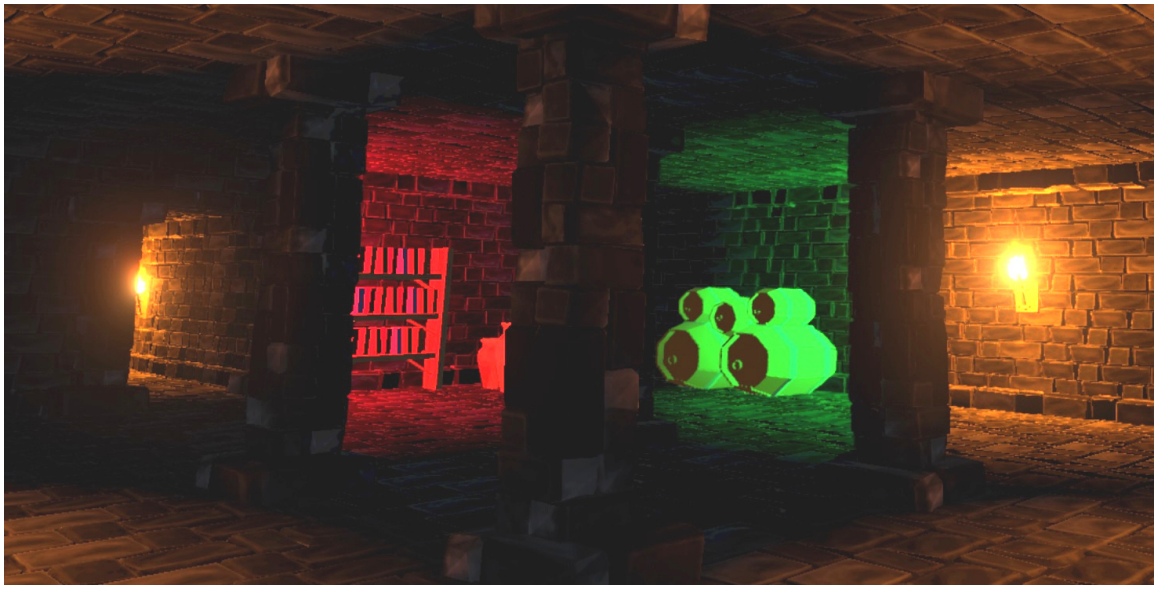


Obr. 4.44: Rozloženie miestností v úrovni desať. Vchodová a východová miestnosť sú označené oranžovou farbou. V úrovni sa tiež nachádzajú dve červené miestnosti na zmätenie hráča. Všetky ostatné miestnosti majú už farby odlišné.



Obr. 4.45: Návrh miestnosti v úrovni desať. V každej miestnosti sú poukladané vymodelované assety. V strede miestnosti sa nachádzajú portály, ktoré umožňujú prechod medzi jednotlivými miestnosťami.

Obr. 4.46: Runy na ktoré natrafí hráč v miestnosti s východom. Každá reprezentuje objekt, ktorý je potrebné nájsť v individuálnych miestnostiach. Jedná sa o jednoduchý štvorec s nanesenou textúrou vytvorenou v programe GIMP.



Obr. 4.47: Pohľad z vchodovej miestnosti na portály, ktoré sú v strede tejto miestnosti. Portál na ľavo vykresľuje pohľad do červenej miestnosti a portál na pravo do zelenej miestnosti.

4.10.12 Úroveň 11 - End Room

Finálna miestnosť. Neobsahuje žiadne hádanky iba poklad, ktorý si hráč po úspešnom prejdení hry patrične zaslúži. V strede miestnosti sa nachádza čarovná teleportačná guľa, obrázok 4.48, ktorá po interakcii s ňou, prenesie hráča na začiatok celej hry.

Obr. 4.48: Koncová miestnosť plná zlata a jedným magickou guľou, ktorý po interakcii s ním prenesie hráča na začiatok hry.

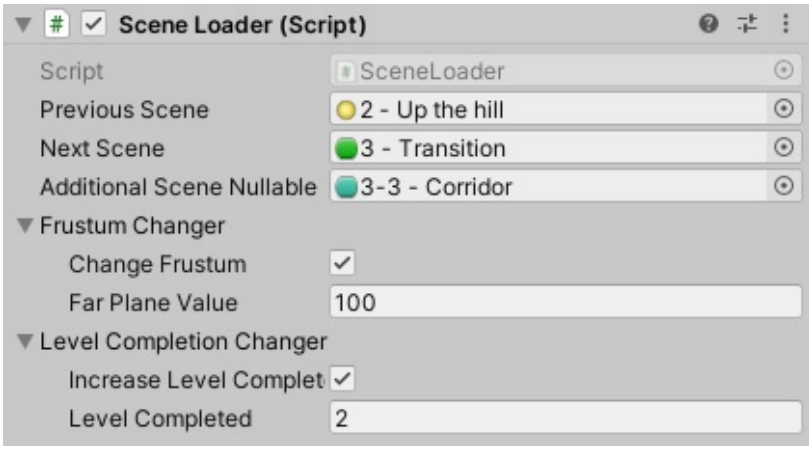
4.10.13 Manažér scén

Jednotlivé úrovne sú navrhnuté v rámci individuálnych scén Unity Enginu, teda jedna scéna zodpovedá jednej úrovni. Okrem scén, reprezentujúcich úrovne boli vytvorené aj scény reprezentujúce prechody medzi danými úrovňami. Tieto scény boli pomenované **X**

- X Corridor a **X - Transition**, kde hodnota X zodpovedá číslu úrovni, na ktoré sú dané prechody napojené. Kompletná cesta medzi úrovňou nula a úrovňou jedna vypadá nasledovne:

- 0 - Starting Room - úroveň 0
- 0 - 1 Corridor - prechod z nultej úrovne do prvej, naviazaný na úroven nula
- 0 - Transition - prechod patriaci úrovni nula, tento prechod spája dva **Corridor**-y, ktoré ho obklopujú
- 1 - 1 Corridor - prechod z nultej úrovne do prvej, naviazaný na úroveň nula jedna
- 1 - Long Tunnel - úroveň 1

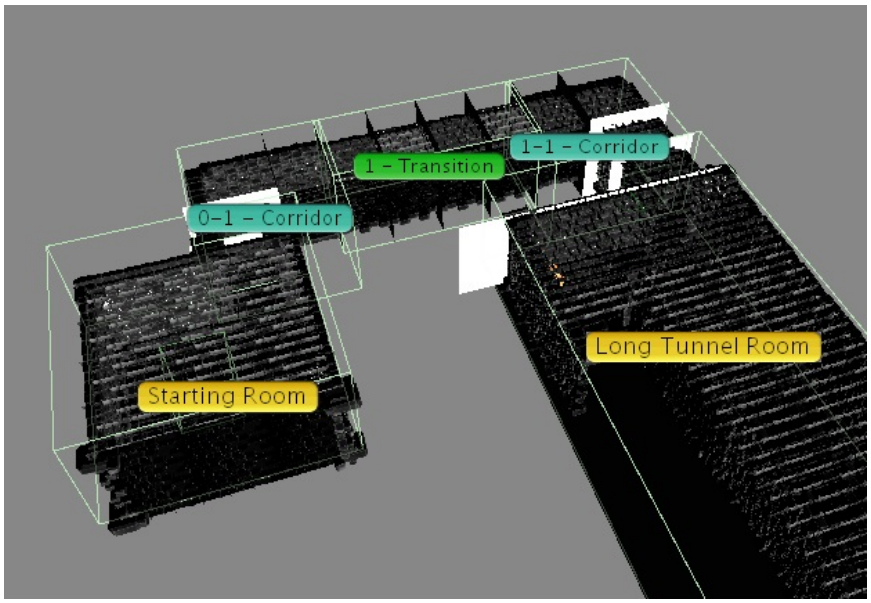
Dôvodom práve takéhoto rozloženia bol fakt, že v každej úrovni sa okrem kamery hráča vykresľujú dodatočné kamery na portály. Mať nahrané všetky úrovne počas celého hrania hry by bolo prakticky nemožné. Pre jednotlivé scény bol vytvorený herný objekt s **BoxColliderom**, ktorý sa stará o ich nahrávania a uvoľňovanie počas behu hry. Tento **BoxCollider** obaluje kompletne celú scénu. Na obrázku 4.50 je možné vidieť túto cestu aj vyznačené **BoxCollidery** jednotlivých scén. Okrem **BoxColliderú** majú tieto herné objekty aj komponentu skript **SceneLoader.cs**, ktorý umožňuje nastaviť, ktoré úrovne sa majú nahrať ako aj iné vlastnosti. Tento skript si získava informáciu, v ktorej scéne sa hráč nachádza a podľa nej potom nahráva okolité scény. Pohľad na túto kompnentu je na obrázku 4.49.



Obr. 4.49: Komponenta **Scene Loader**. Umožňuje nastaviť, ktoré dodatočné scény sa majú nahrať. **Previous Scene** predstavuje predošlú scénu odkiaľ hráč prišiel, **Next Scene** predstavuje nasledujúcu scénu. **Additional Scene Nullable** predstavuje špeciálny prípad keď je potrebné nahrať okrem predošlej a nasledujúcej scnény aj scénu inú. Okrem nahrávania scén umožňuje komponenta meniť frustum kamery hráča a taktiež zaznamenávať posledný úspešne prejdený level.

Keďže niektoré úrovne obsahujú aj sekundárny cieľ, bolo nutné si udržiavať informáciu o tom, ktorá úroveň bola úspešne prejdená. Každá scéna obsahuje herný objekt so skriptom **OnLevelLoad.cs**, ktorý sa pozrie na najvyššiu dosiahnutú úroveň a podľa nej nahrá scénu ako novú alebo už prejdenú. Toto bolo nutné implementoať pre prípad ak by sa hráč rozhodol otočiť naspäť a navštíviť predošlé úrovne. Vďaka tomuto spôsobu implementácie scén bolo

možné rozšíriť užívateľské rozhranie o špecializované menu, ktoré umožňuje hráčovi výber konkrétnej úrovne a následne nahranie zvolenej úrovne.



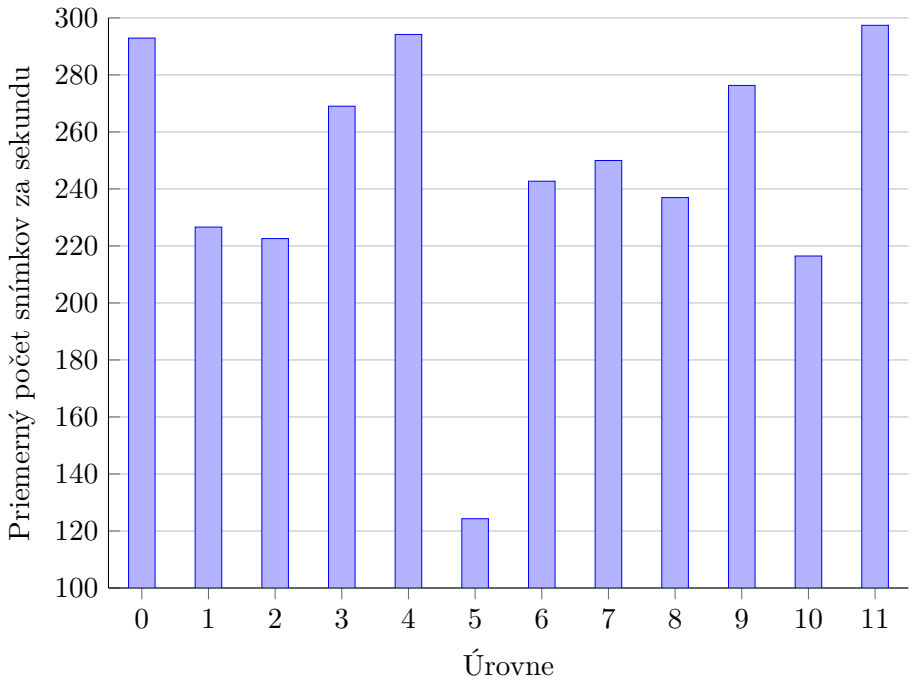
Obr. 4.50: Kompletná cesta medzi scénami s vyznačenými príslušnými **BoxColliderami**.

Kapitola 5

Vyhodnotenie aplikácie

Aplikácia bola vyvíjaná na stolovom počítači s procesorom **AMD Ryzen 3700X**, grafickou kartou **Nvidia GeForce 2070 Super** a pamäťou **16 GB RAM**. Ako vykresľovací reťazec bol spočiatku zvolený univerzálny vykresľovací reťazec (URP). Pri pripojení virtuálnej reality s využitím balíkov **XR Interaciton Toolkit** a **XR Plug-in Management** sa aplikácia navonok tvárila v poriadku a proces vývoja nebol značne ovplyvnení. Problém nastal až pri vykresľovaní portálových efektov, kde rovnako ako univerzálny vykresľovací reťazec, tak aj HD vykresľovací reťazec pri spustení aplikácie nesprávne inicializovali stereoskopické vykresľovanie. V oboch prípadoch a na viacerých verziách Unity Enginu, sa nesprávne nenastavovala hodnota **Camera.stereoEnabled**, to znamenalo že vo fragment shadery **PortalVR** shader programu nebolo možné medzi jednotlivými očami za pomoci premennej **unity_StereoEyeIndex** prepínať a nanášať správne textúry. Ako vykresľovací reťazec bol nakoniec zvolený vstavaný vykresľovací reťazec, teda Built-in render pipeline nastavený na Multi-pass vykresľovanie. Prepnutím vykresľovacieho reťazca v polovici vývoja aplikácie znamenalo, že všetky materiály vytrvorené v URP bolo potrebné prerobiť aby boli kompatibilné s novým vykresľovacím reťazcom. Ako už bolo spomenuté v kapitole 4.2, toto Multi-pass vykresľovanie by malo zabezpečiť kompatibilitu aj z ostatnými headsetmi v prípade ak by bola snaha aplikáciu rozširovať.

Keďže na spustenie aplikácie je potrebný VR headset, bola aplikácia spúštaná a testovaná iba na vyššie spomenutom vývojovom stroji. Aplikácia je prispôsobená na hranie v sede, testy z tohoto dôvodu neboli zamerané primárne na efekty a pohyb v úrovniach ale skôr na rýchlosť vykresľovania počas samotného hrania, keďže v jednotlivých úrovniach sa vykresľuje vždy odlišný počet kamier. Aplikácia bola rozšírená tak, aby si o jednotlivých úrovniach udržiavala priemerný počet snímkov za sekundu a zapisovala si tento počet do príslušných súborov. Na grafe 5.1 sú priemerné hodnoty snímkov za sekundu v jednotlivých úrovniach počas ich prechádzania. Samotný Oculus Rift má však pevne nastavenú vertikálnu synchronizáciu bez možnosti vypnutia. V prípade merania rýchlosti vykresľovania aplikácie bolo nutné merať aplikáciu z nenasadeným headsetom. To spôsobilo že dané vykresľovanie nebolo orezané na limit zariadenia, teda 90 snímkov za sekundu.



Obr. 5.1: Graf zobrazujúci priemerný počet snímkov za sekundu v individuálnych úrovniach. Najväčší dopad na počet snímkov mala úroveň 5. Je to zo všetkých najväčšia úroveň. Úrovne 0, 4, 9, 11 dosahovali najvyššieho počtu snímkov za sekundu. Sú to miestnosti, kde sa nevykresľoval ani jeden portál. Úrovne, kde sa vykresľoval vždy iba jeden portál dosahoval priemerný počet snímkov za sekundu vyšší ako 220. Sú to úrovne 1, 2, 3, 6, 7, 8. V úrovni 10 sa počet vykresľovaných portálov počas behu aplikácie dynamicky menil. V úrovni je totiž možné sa pozerať na dva portály súčasne. Z toho dôvodu je logicky FPS o niečo menšie.

Kapitola 6

Možné rozšírenia aplikácie

Výsledná aplikácia je hrateľná, avšak nie je ani zďaleka dokonalá a je veľa oblastí, v ktorých by bolo možné ju zdokonaliť. Pri plynulom priechode portálu dochádza k občasnému prebliknutiu obrazovky. To je spôsobené tým, ze samotná teleportácia sa vykonáva na základe pozície postavy hráča a nie kamery. Tá totiž môže mať nastavený mierny offset, ktorý je ovplyvnený samotným sledovaním zariadenia.

Spôsob akým bola aplikácia vyvíjaná ju veľmi limituje na kombináciu headsetu s ovládačom Xbox. Možným rozšírením by bolo pridanie funkcionality prvotne pre ovládače Oculus Touch. S ich podporou by bolo možné s jednotlivými modelmi manipulovať, zdvíhať, prenášať. Táto nová funkcionalita by otvorila dvere novým spôsobom návrhu jednotlivých úrovní, komplexnejším hádankám.

Po audio-vizálnej stránke je taktiež neskutočne veľa možných vylepšení. Aplikácia síce využíva pomerne dosť modelov avšak modely sú pomerne jednoduché. Ďalším možným rozšírením by bolo pridanie kvalitnejších, prepracovanejších a zároveň aj nových modelov. Okrem pridania týchto modelov by bolo ďalej možné ich individuálne na animovať a pridať im rôznu funkcionalitu. Po čisto zvukovej stránke aplikácia využíva zvukové efekty dostupné zadarmo na internete. Keďže nie sú efekty priamo vyvíjané pre zvolenú aplikáciu, nie vždy plne zapadnú a dodajú práve audio vnem, o ktorý bola snaha. Možnosťou rozšírenia by bolo vytvoriť si vlastné efekty, ktoré by presne reprezentovali jednotlivé objekty v scéne a akú hudbu alebo efekt majú prehrať pri interakcii s nimi.

Kapitola 7

Záver

Cieľom diplomovej práce bolo vytvoriť aplikáciu, ktorá bude schopná demonštrovať vykresľovanie neeuklidovských priestorov. Úvodná časť práce bola venovaná štúdiu jednotlivých hardvérových zariadení umožňujúce zapojenie sa do virtuálneho priestoru a taktiež samotným neeuklidovským priestorom. Bolo potrebné si taktiež naštudovať ako daného neeuklidovského priestoru alebo efektu dosiahnuť v kombinácii s virtuálnou realitou. Do úvahy spadali tri možnosti: Unreal Engine, Unity Engine, OpenGL. Poznatky získané štúdium daného problému viedli k voľbe Unity Enginu. Dôvodom sa stala dostupnosť, podpora ale hlavne jednoduchá integrácia virtuálnej reality pomocou špecializovaných balíčkov.

V ďalšej časti práce bol popísaný zvolený engine, jeho výhody a princípy práce v ňom ako aj samotná implementácia aplikácie a jej systémov. V práci boli implementované vlastné systémy na pohyb hráča a interakciu s objektami. Práca popisuje implementáciu portálov ako aj využitie stencilových masiek, na docielenie žiadaného neeuklidovského priestoru, efektu. Samostatná podkapitola je venovaná návrhu užívateľského rozhrania aplikácie a problémom ktoré sa pri jeho návrhu vyskytli. Po implementácii užívateľského rozhrania je časť práce venovaná estetike samotnej aplikácie. Venuje pozornosť zvolenému grafickému štýlu ako aj osvetleniu a jednotlivým zvukovým efektom použitých v aplikácii. Pokračuje sa rozdelením jednotlivých neeuklidovských efektov do samostatných scén, úrovní a ich správa. Počas behu aplikácie sú vždy nahrané maximálne štyri úrovne, tá, v ktorej sa hráč nachádza a jej dve susedné úrovne, prípadne ešte explicitne definovaná dodatočná úroveň. V práci bolo vytvorených desať unikátnych úrovní plus počiatočná úroveň a koncová úroveň. Jednotlivé úrovne slúžia na demonštráciu neeuklidovských priestorov, efektov, s využitím implementovaných portálov, stencilových masiek alebo samotným rozpoložením miestností v úrovni. Každá úroveň bola navrhnutá tak, aby mala jasný začiatok a cieľ kam sa má hráč dostať aby postúpil na ďalšiu úroveň. Výsledkom práce je spustiteľná aplikácia, ktorej základný koncept je veľmi podobný počítačovým hrám ako hra Antichamber.

Na záver sa práca venuje jednoduchému testovaniu hotovej aplikácie na vývojovom stroji a meraniu rýchlosti vykresĺovania implementovaných efektov v jednotlivých úrovniach. Samotná kapitola je venovaná aj stručnému popisu potencionálnych rozšírení aplikácie. Najzaujímavejším by však bolo rozšíriť aplikáciu o ovládače umožňujúce sledovanie vo virtuálnej realite ako aj návrh nových úrovní a pridanie viacerých objektov, s ktorými by hráč mohol rôzne interagovať a manipulovať.

Literatúra

[1] Barnard, D. *History of VR - Timeline of Events and Tech Development* [online]. August 2019 [cit. 2021-19-05]. Dostupné z: https://virtualspeech.com/blog/history-of-vr.

[2] Brown, M. *Exploring the magic behind the HTC Vive controller* [online]. 2016 [cit. 2021-19-05]. Dostupné z: https://www.vrheads.com/exposing-magic-behind-htc-vive-controller.

[3] Buckley, S. *This Is How Valve's Amazing Lighthouse Tracking Technology Works* [online]. 2015 [cit. 2021-19-05]. Dostupné z: https://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768.

[4] Heilig, M. L. *Sensorama Simulator*. United States patent US 3050870A. January 10 1961 [cit. 2021-19-05]. Dostupné z: https://web.opendrive.com/api/v1/download/file.json/Ml8xNTA4NjQyOTJf?inline=1.

[5] Hulton, T. *Portal Rendering with Offscreen Render Targets* [online]. 2015 [cit. 2021-19-05]. Dostupné z: http://tomhulton.blogspot.com/2015/08/portal-rendering-with-offscreen-render.html.

[6] Leone, M. *The Making of PlayStation VR* [online]. 2016 [cit. 2021-19-05]. Dostupné z: https://www.polygon.com/2016/3/9/11174194/the-making-of-playstation-vr.

[7] Malventano, A. *SteamVR HTC Vive In-Depth – Lighthouse tracking system dissected and explored* [online]. 2016 [cit. 2021-19-05]. Dostupné z: https://pcper.com/2016/04/steamvr-htc-vive-in-depth-lighthouse-tracking-system-dissected-and-explored/2/.

[8] Richards, M. *Ivan Sutherland's experimental 3-D display* [online]. [cit. 2021-19-05]. Dostupné z: https://www.computerhistory.org/revolution/input-output/14/356/1830.

[9] Robertson, A. a Zelenko, M. *An oral history of a technology whose time has come again* [online]. The Verge, 2016 [cit. 2021-19-05]. Dostupné z: https://www.theverge.com/a/virtual-reality/oral_history.

[10] Skrodzki, M. *Illustrations of non-Euclidean geometry in virtual reality* [ResearchGate]. 2020. [Online] [cit. 2021-19-05]. Dostupné z: https://www.researchgate.net/publication/343441981_Illustrations_of_non-Euclidean_geometry_in_virtual_reality.

[11] Srinivasiah, R. *How to maximize AR and VR performance with advanced stereo rendering* [Unity3d, blog]. 2017 [cit. 2021-19-05]. Dostupné z: https://blogs.unity3d.com/2017/11/21/how-to-maximize-ar-and-vr-performance-with-advanced-stereo-rendering/.

[12] Sutherland, I. E. The Ultimate Display. In: *Proceedings of the Congress of the Internation Federation of Information Processing (IFIP)*. 1965, s. 506–508.

[13] Sutherland, I. E. A head-mounted three dimensional display. In: *Fall Joint Computer Conference*. 1968, s. 757–764.

[14] The Oculus team. *The Oculus Rift, Oculus Touch, and VR games at E3* [online]. 2015 [cit. 2021-19-05]. Dostupné z: https://www.oculus.com/blog/the-oculus-rift-oculus-touch-and-vr-games-at-e3/.

[15] The Oculus team. *Oculus Rift and Rift S minimum requirements and system specifications* [online]. 2020 [cit. 2021-19-05]. Dostupné z: https://support.oculus.com/248749509016567/.

[16] Thompson, S. *Motion Sickness in VR: Why it happens and how to minimise it* [virtualspeech, blog]. 2020. [Online] [cit. 2021-19-05]. Dostupné z: https://virtualspeech.com/blog/motion-sickness-vr.

[17] Unity Technologies. *Input System* [online]. 2021 [cit. 2021-19-05]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/index.html.

[18] Unity Technologies. *Unity User Manual (2020.1)* [online]. 2021 [cit. 2021-19-05]. Dostupné z: https://docs.unity3d.com/2020.1/Documentation/Manual/index.html.

[19] USC Hugh M. Hefner Moving Image Archive. *Inventor in the field of virtual reality* [online]. 2021 [cit. 2021-19-05]. Dostupné z: https://www.uschefnerarchive.com/morton-heilig-inventor-vr/.

[20] Valve. *Valve Index* [online]. 2020 [cit. 2021-19-05]. Dostupné z: https://www.valvesoftware.com/en/index/.

[21] Zucconi, A. *Impossible Geometry: Non-Euclidean Cubes* [online]. 2015 [cit. 2021-19-05]. Dostupné z: https://www.alanzucconi.com/2015/12/09/3873/.

Príloha A

Obsah priloženého DVD

Štruktúra priloženého DVD je nasledovná:

- **Blender**
  - modely a textúry vytvorené v programe Blender 2.92.0
- **Build**
  - spustiteľná aplikácia
- **Source**
  - zdrojové súbory aplikácie
- **TextSource**
  - zdrojové súbory textovej časti diplomovej práce
- **Neeuklidovske_vykreslovanie_vo_VR.pdf**
  - text diplomovej práce
- **Neeuklidovske_vykreslovanie_vo_VR.mp4**
  - demonštračné video aplikácie
- **README.txt**
  - bližší popis štruktúry práce na DVD

Príloha B

Vytvorené modely